IBM WebSphere Commerce Payments for Multiplatforms

Administrator’s Guide

Version 3.1

IBM WebSphere Commerce Payments for Multiplatforms

Administrator’s Guide

Version 3.1

Note

Before using this information and the product it supports, be sure to read the general information under Appendix E, “Notices” on page 135.

Third Edition (July 2002)

This edition applies to Version 3.1.3 of IBM® WebSphere® Commerce Payments for Mulitiplatforms and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

This guide provides information about WebSphere Commerce Payments. It is for system administrators and anyone responsible for installing, configuring and maintaining WebSphere Commerce Payments.

Note:
IBM WebSphere Commerce Payments for Multiplatforms (hereafter called WebSphere Commerce Payments) was previously known as Payment Manager. Starting with version 3.1.3, the payments application was renamed to WebSphere Commerce Payments and references to the product were changed throughout this document. References to the former product may still appear in this document and apply to earlier releases of the product.

What's new for this release

For ease of migration, release 2.2.x cassettes will work with the 3.1.3 framework.

New instructions for migrating configuration and transaction data to the latest version of WebSphere Commerce Payments have been added. See the *IBM WebSphere Commerce Payments for Multiplatforms Installation Guide* for more information.

The *IBM WebSphere Commerce Payments Administrator's Guide* reflects changes for WebSphere Commerce Payments Version 3.1. Major changes include:

IBM WebSphere Application Server 4.0.x support

Support for WebSphere Application Server Version 4.0.x is now provided. WebSphere Application Server Versions 2.03, 3.0.2 and 3.5.4 are no longer supported. The WebSphere Application Server handles the starting and stopping of the WebSphere Commerce Payments Servlet. Also, your Web server (e.g., IBM HTTP Server) and database must be supported by WebSphere Application Server 4.0.x.

Because WebSphere Application Server Version 4.0.x is compliant with the Java 2 Enterprise Edition (J2EE) specification the WebSphere Commerce Payments directory structure has changed, along with the configuration of WebSphere Application Server relative to that of Payment Manager Version 3.1.0 and earlier releases.

- To configure WebSphere Application Server, WebSphere Commerce Payments now makes use of Web archive (WAR) and enterprise archive (EAR) files. WAR files (.war) are used to package Web modules. A Web module can represent a stand-alone Web application, or it can be combined with other modules (for example, EJB modules) to form a J2EE application. The J2EE application can then be installed and run in a WebSphere application server. An Enterprise Archive file represents a J2EE application that can be deployed in a WebSphere application server. EAR files (.ear) are standard Java archive files and have the file extension `.ear`.
  - This change is generally transparent to you unless you are writing payment cassettes for use with WebSphere Commerce Payments. If you are writing cassettes, refer to WebSphere Commerce Payments for Multiplatforms Cassette Kit Programmer's Guide for more information about how to ensure that your cassette properly

configures WebSphere Application Server. The document is packaged with the WebSphere Commerce Payments cassette toolkit available from:

http://www.ibm.com/software/webservers/commerce/payments /download.html.

Some files that you may have seen in the WebSphere Commerce Payments installation directory prior to this release are now moved into the EAR file structure and deployed to a WebSphere Application Server directory. The following files are now located in the EAR file:

```
Payments_installdir\web\* (for framework and
 installed cassettes)
Payments_installdir\eTillCal.zip
Payments_installdir\eTillClasses.zip
Payments_installdir\eTillCustomOfflineCassetteClasses.zip
Payments_installdir\eTillOfflineCardCassetteClasses.zip
Payments_installdir\eTillUI.zip
Payments_installdir\eTillxml4j209.jar
Payments_installdir\ibmjsse.jar
Payments_installdir\WCSRealm.jar
```

The cassette files that are moved into the EAR are as follows:

```
Payments_installdir\CassetteName.zip (i.e.-eTillSETClasses.zip)
Payments_installdir\web\* (that pertain to the cassette)
```

WebSphere administrative console

The WebSphere administrative console is a tool used to start and stop the WebSphere Commerce Payments Servlet, and it allows you to configure application server configuration files. Other common tasks that are performed in the administrative console include assembling the modules of the application, setting deployment descriptor properties, and changing WebSphere Commerce Payments initialization parameters using the Application Assembly Tool.

Database product support

- IBM DB2 Universal Database (UDB) and Oracle databases continue to be supported. Supported versions are UDB Version 7.2 (FixPak 4 or higher) and Oracle 8i (8.1.7) or later (up to but not including 9i). Lower versions of these database products are no longer supported: UDB Version 6.1.x., Oracle 8.0.x.
- Support for Microsoft SQL Server is no longer provided. Users of Microsoft SQL Server can migrate database tables to UDB using the migrateDB utility shipped with WebSphere Commerce Payments. For more information on migrating MS SQL databases to UDB, refer to the WebSphere Commerce Payments Installation Guide.

European currency conversion

Previously in the WebSphere Commerce Payments user interface, the amount of an order was specified according to the specific European country currency, and the currency value was specified as Euro. In version 3.1.2, currency value for the amount field and the currency field are reversed. For example, if 20.00 French Francs was specified as the amount of an order in a previous version, then WebSphere Commerce Payments will convert this amount to 1.94 Euro, and the currency will be specified as French Francs.

Tivoli Ready™ support
This version of WebSphere Commerce Payments does not provide Tivoli Ready™ support any longer.

UTF8 Compliance
In previous WebSphere Commerce Payments versions, if an administrator were to specify Korean as the language for WebSphere Commerce Payments, and they wanted to view merchants' names that were in Japanese characters, those characters would not be visible. Now that WebSphere Commerce Payments is UTF8 compliant, the administrator can specify a particular language in WebSphere Commerce Payments and still be able to view other language characters.

Accessibility Compliance
WebSphere Commerce Payments is compliant with user accessibility standards such as having the ability to use keyboard shortcuts, screen-reader software, digital speech synthesizers and other assistive devices.

Test Cassette
The Test Cassette is no longer supported. Instead, use the OfflineCard or CustomOffline cassettes to perform payment processing that does not require financial network connectivity.

Understanding WebSphere Commerce Payments terms

The following key terms and concepts are used throughout this book:

Buyer A person who uses software (such as, an electronic wallet) to purchase products and services from an Internet merchant. Also known as shopper, or cardholder.

Framework
The structure of the WebSphere Commerce Payments that allows for different merchant servers using different payment systems to issue the same generic commands to the WebSphere Commerce Payments and use the same generic data. The WebSphere Commerce Payments uses protocol-specific cassettes to translate the generic calls to protocol-specific messages.

Merchant administrator
The merchant representative with administrator access to all merchant functions.

Merchant server
A Web server that offers catalogued shopping. Examples are Lotus® Domino.Merchant™, IBM WebSphere Commerce Business Edition and IBM WebSphere Commerce Studio, Business Developer Edition.

Payment cassette
Software that implements an electronic payment protocol. Examples are the SET cassette and the CyberCash cassette.

Payment gateway
For many payment types, the entity that handles transactions between a merchant and a financial institution.

Payments administrator
The person with administrator access to all WebSphere Commerce Payments functions.

Realm WebSphere Commerce Payments authenticates users through the use of realms. A realm is registry of users along with a single method of authenticating those users (for example, a user's name and password). Examples of realm types include LDAP realms and operating system realms. A user must be defined in a realm before being granted access to resources in that realm.

Wallet For many payment types, software that enables a buyer to make approved payments to merchants over the Internet.

JDK Java Development Kit

UDB DB2 Universal Database™ product

J2EE J2EE is designed to support applications that implement enterprise services for customers, employees, suppliers, partners, and others who make demands on or contributions to the enterprise. This can be a single module or a group of modules packaged into an .ear file with a J2EE application deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers.

WAR file
A Web Archive (WAR) file is a Java archive file used to store one or more of the following:

- Servlets
- JavaServer Pages (JSP) files
- Utility classes
- Static documents, such as HTML files, images and sound
- Client-side applets, beans and classes
- Descriptive meta-information

Its standard file extension is `.war`. WAR files are used to package Web modules.

EAR file
An Enterprise Archive file represents a J2EE application that can be deployed in a WebSphere application server. EAR files are standard Java archive files and have the file extension `.ear`.

Note: References to *workstation* and *multiplatforms* refer to Windows®, AIX®, Solaris and Linux®.

Conventions in this document

Table 1. Document conventions

Boldface	Indicates the name of the item you need to select, the name of a field, or a string you must enter.
Italics	Indicates book titles or variable information that must be replaced by an actual value.
`Monospace`	Indicates an example, a portion of a file, or a previously entered value.

For more information

All documents are available on the WebSphere Commerce Payments CD-ROMs in Portable Document Format (PDF). On iSeries™ systems, the documentation is compressed into a save file and is only available after WebSphere Commerce Payments has been installed. In addition to this manual, you can reference the following related Web sites and documents:

- **http://www.ibm.com/software/websphere/paymgr/support/index.html** provides current WebSphere Commerce Payments technical information and links to the latest WebSphere Commerce Payments documentation.
- **http://www.ibm.com/software/websphere/appserv/library.html** provides documentation links for IBM WebSphere Application Server.
- **http://www.ibm.com/software/data/pubs/index.html** provides documentation links for IBM Universal Database.
- **http://www.ibm.com/software/commerce/connect** provides information on:
  - Payment Processing Services that support WebSphere Commerce Payments. Specifically, this Web site directs you to the Financial Institutions that have established payment gateways to handle your WebSphere Commerce Payments payment processing needs.
  - Payment cassettes that are available for use with WebSphere Commerce Payments. This site contains connectivity and functionality information, as well as links to ordering information for the available cassettes.
- **http://www-4.ibm.com/software/webservers/appserv/doc/v40/aee/index.html** provides a link to documentation on J2EE specification.
- **http://www.software.ibm.com/websphere/paymgr/download.html** provides information on WebSphere Commerce Payments cassette development.
- The *IBM WebSphere Commerce Payments for Multiplatforms Install Guide* provides information for installing, migrating, and starting the WebSphere Commerce Payments.
- The *IBM WebSphere Commerce Payments for Multiplatforms Programmer's Guide and Reference* provides information for programmers who develop applications that interface with the WebSphere Commerce Payments.
- The *WebSphere Commerce Payments Cassette for SET Supplement* provides information about using the SET protocol with WebSphere Commerce Payments to process electronic payments.
- The *WebSphere Commerce Payments Cassette for CyberCash Supplement* provides information about using the CyberCash protocol with WebSphere Commerce Payments to process electronic payments.
- The *WebSphere Commerce Payments Cassette for VisaNet Supplement* provides information about performing payment processing and authorizing and settling payments.
- The *WebSphere Commerce Payments Cassette for BankServACH Supplement* provides information on supporting online electronic check payments.

Chapter 1. WebSphere Commerce Payments overview

What is WebSphere Commerce Payments?

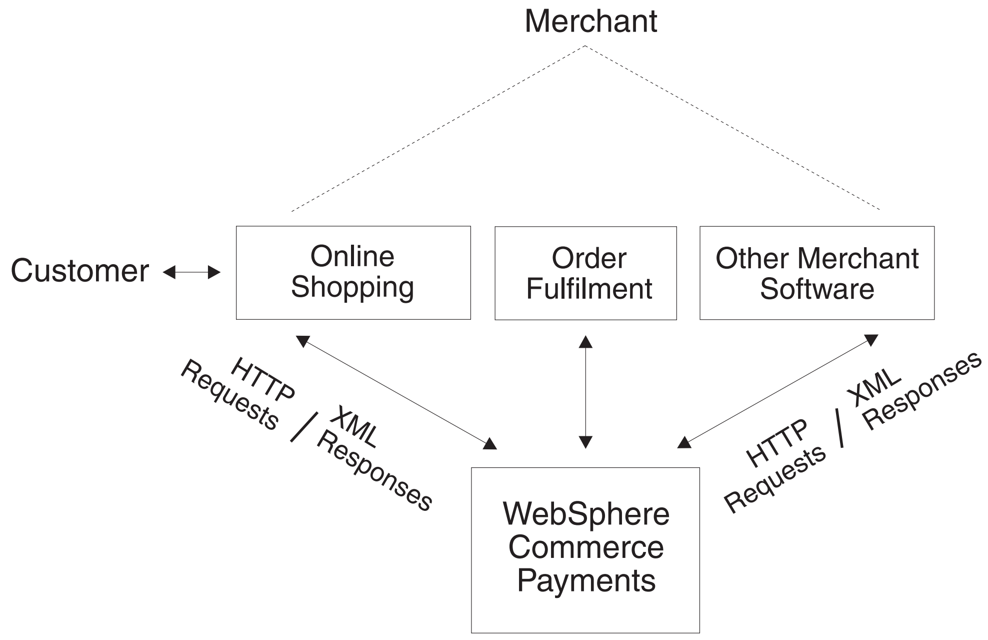
In the beginning, the Internet was designed to provide an open network between scientists. Today, it has evolved into an important medium for online commerce. While electronic commerce is quickly emerging as a leading business model, transacting business on the Internet presents two major security challenges:

1. Buyers are wary of launching their vital payment card data into the unknown.
2. Merchants have been unable to easily confirm the identity of buyers using payment cards.

WebSphere Commerce Payments bridges the security issues and provides secure, electronic payment processing to Internet merchants. Based on open, standards-based technology, WebSphere Commerce Payments works with payment *cassettes* (see “WebSphere Commerce Payments’s answer to multiple payment types” on page 4) to support multiple payment protocols, including:

- SET (Secure Electronic Transaction™), an industry-standard transport and messaging protocol using encrypted certificates for highly secure payment processing
- Merchant Initiated SET (MIS), a SET™ extension that can be used by merchants to accept credit card information from buyers using any method other than SET
- Cassette for CyberCash, which allows users of WebSphere Commerce Payments to access CyberCash, Inc. CashRegister Credit Card Service
- VisaNet, a provider of worldwide telecommunications data and payment processing that has the ability to authorize and settle payments.
- BankServACH, a payment gateway that interfaces with the Automated Clearing House Network (ACH) to support online electronic check payments.
- third-party payment cassettes written for the WebSphere Commerce Payments.

Simply put, WebSphere Commerce Payments integrates with merchant software systems (for example, order fulfillment and online shopping) and provides cash register-like functionality to manage payment processing. The buyer never interacts with WebSphere Commerce Payments directly because WebSphere Commerce Payments sits *behind* the Internet merchant’s storefront, receiving payments and processing those payments with banks and other financial institutions (see “The WebSphere Commerce Payments and e-commerce entities” on page 6). Figure 1 illustrates this concept:


Merchant
Customer
Online Shopping
Order Fulfilment
Other Merchant Software
HTTP Requests / XML Responses
HTTP Requests / XML Responses
WebSphere Commerce Payments

At the bottom of the figure, the WebSphere Commerce Payments is shown as a server (or manager) that handles requests from the merchant software (illustrated at the top right). Note that the WebSphere Commerce Payments can service multiple pieces of merchant software.

A hosted WebSphere Commerce Payments

WebSphere Commerce Payments is designed to support:

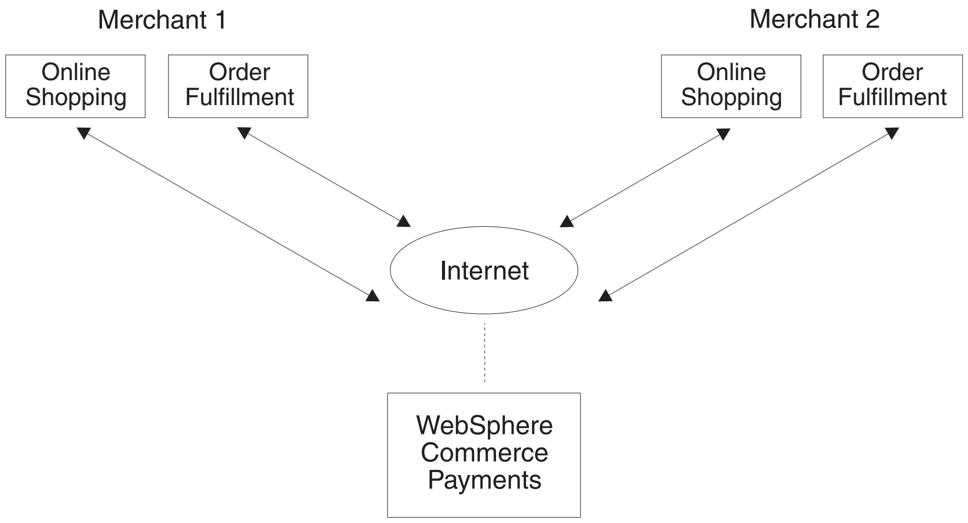
- Different types of merchant software
- Multiple merchants in a hosted environment (that is, multiple merchants using the same WebSphere Commerce Payments)

From the point of view of merchant software, each merchant logically has its own WebSphere Commerce Payments and is completely unaware that other merchants are using the same WebSphere Commerce Payments. WebSphere Commerce Payments not only enables this logical separation and dedication of resource, it also enforces security and protects the merchants from each other through authentication and access control. For example, each merchant can view and take action only upon its own data (that is, the merchant can perform payment actions *only* on its own financial objects).

In addition, the WebSphere Commerce Payments enforces roles such that each user is presented with a different view based on the user's role (for example, from the perspective of a Payments administrator versus a Merchant administrator). Even within the merchant organization, WebSphere Commerce Payments enables the notion of different roles so that the merchant can police its own users. For example, a clerk may be restricted to operations such as approving an order, while only a merchant administrator can modify a relationship with a financial institution.

The following scenario illustrates the hosted WebSphere Commerce Payments concept. In this figure, one WebSphere Commerce Payments hosts two merchants, each with multiple software systems that are supported by the WebSphere

Commerce Payments: an online shopping center and an order fulfillment system.



WebSphere Commerce Payments’s multi-payment Framework

WebSphere Commerce Payments’s multi-payment Framework expands the audience for a commerce Web site by enabling merchants to provide as many payment methods as the audience may need. The multi-payment Framework helps merchants accept multiple payment methods, customize specific payment methods with varied financial institutions, and adapt to rapidly changing business requirements and technology by easily adding more payment type options as they emerge.

e-Commerce and multiple payment options

Business on the Internet, especially making and receiving payments, creates the need for a payment processing system that accommodates multiple payment types in a secure environment. Today, buyers can browse catalogs, order items, and pay online using credit cards, debit cards, stored-value smart cards, micropayments, and other payment options. An order received over the Internet can come from virtually anywhere in the world, and many different payment schemes are being introduced in various countries.

The Internet is driving the need for multiple payment types in several ways:

- Internet-based e-commerce is growing at a rapid pace, creating a market demand for payments of all types.
- The Internet brings together payers and payees from all over the world in new combinations. Many of the payers and payees come from different cultural, legal and business practice backgrounds and already have access to different payment types. To do business on the Internet, they need to support each other’s payment types.
- As an open public network, the Internet requires security techniques that are not needed to the same degree offline or in private networks.

These factors force Internet payment methodologies to evolve rapidly to meet different combinations of user needs. The result is a wide variety of payment types.

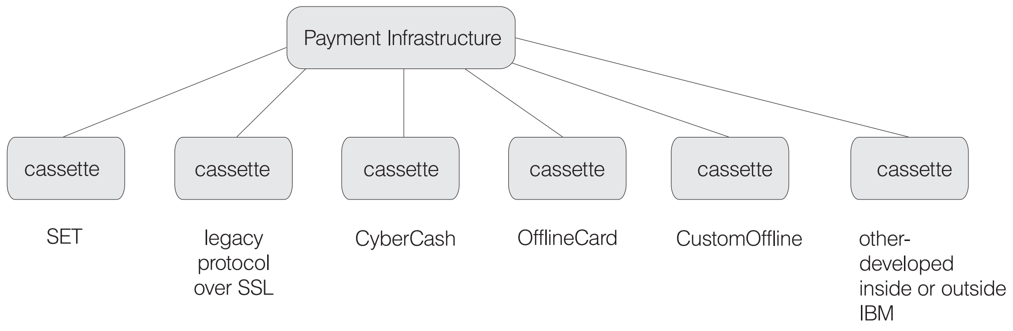
WebSphere Commerce Payments's answer to multiple payment types

The WebSphere Commerce Payments implements a multi-payment Framework architecture that provides a flexible and extensible way to accommodate payment requirements on the Internet for merchants who need to accept multiple payment methods. The multi-payment Framework separates payment management, the *Framework*, from specific payment types, the software *cassettes*, so that each can evolve independently.

The WebSphere Commerce Payments provides a plug-in architecture whereby software cassettes for each payment type attach to the payment Framework. The Framework provides the generic infrastructure functions required for making and receiving payments using any payment type.

Payment cassettes are software applications that conform to the data flow and control conventions of the WebSphere Commerce Payments Framework. Each payment cassette contains the implementation of specific payment methods and protocols.

As shown in the following diagram, WebSphere Commerce Payments currently provides a cassette for the SET protocol, CyberCash, VisaNet, BankServACH and two Offline cassettes. The Offline cassettes are provided with the framework. Other payment protocol cassettes are planned for the future.



Cassettes can be written by IBM or by third-party payment system implementors. IBM supports cassette development and offers detailed instruction to developers interested in writing their own payment cassettes. For more information on cassette development, see **http://www.software.ibm.com/commerce/payment/download.html** and the *Cassette Kit Programmer's Guide for WebSphere Commerce Payments*.

For details on how a particular software cassette is accessed through this generic Framework, see the appropriate cassette-specific supplement:

- The *WebSphere Commerce Payments for SET Supplement* provides information about using the SET protocol with the WebSphere Commerce Payments Framework to process electronic payments.
- The *WebSphere Commerce Payments for CyberCash Supplement* provides information about using the CyberCash protocol with the WebSphere Commerce Payments Framework to process electronic payments.
- The *WebSphere Commerce Payments Cassette for BankServACH Supplement* provides information about the BankServ payment gateway, which uses the HTTPS protocol.
- The *WebSphere Commerce Payments Cassette for VisaNet Supplement* provides information about the VisaNet protocol with the WebSphere Commerce Payments Framework to process electronic payments.

For more information about the CustomOffline Cassette, see Appendix B, “CustomOffline cassette supplement” on page 73. For more information about the OfflineCard Cassette see Appendix A, “OfflineCard cassette supplement” on page 45.

The WebSphere Commerce Payments user interface

The WebSphere Commerce Payments provides a user interface that can be used by Payments administrators and individual merchants using hosted WebSphere Commerce Paymentss to:

- Configure the WebSphere Commerce Payments
- Perform routine payment processing tasks in a payment-neutral way:
  - Approve Orders
  - Deposit Payments
  - Settle Batches
  - Issue Credits
  - View Daily Batch Totals

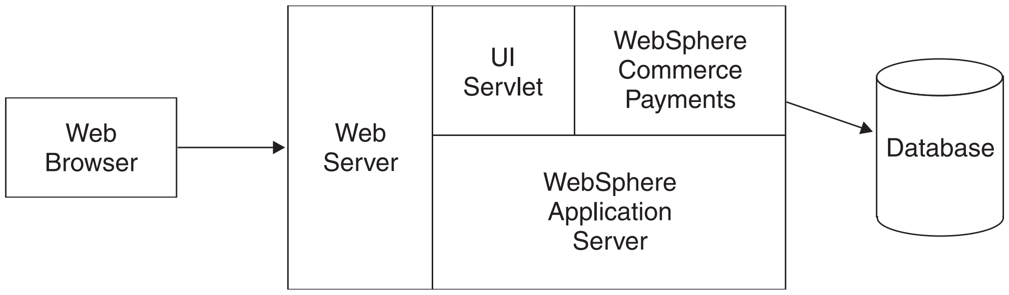
The WebSphere Commerce Payments user interface is browser-based and can be accessed remotely and securely using the Secure Sockets Layer (SSL) capabilities of your Web browser.

WebSphere Commerce Payments: internal components

Having examined the WebSphere Commerce Payments and how it interacts with *external* components, it is useful to look inside the WebSphere Commerce Payments and define its *internal* components:

- Web server configured with WebSphere Application Server
- WebSphere Commerce Payments Servlets
- User Interface servlet
- Database

The following figure illustrates the internal components of the WebSphere Commerce Payments:



The WebSphere Commerce Payments Servlet is the major component for WebSphere Commerce Payments. The Servlet is designed to work with WebSphere Application Server – a product that provides a common servlet environment for all platforms.

By using a Web server-based architecture, WebSphere Commerce Payments utilizes the Web server’s ability to handle HTTP requests while at the same time leveraging its many security features. For more information on Web server security features and SSL, see “Protecting the Web server using SSL” on page 31.

The remainder of this section further defines the WebSphere Commerce Payments Servlet as well as their ancillary support products:

- Web server
- WebSphere Application Server
- Database

Web server

Why a Web server? As mentioned earlier in this chapter, WebSphere Commerce Payments processes incoming HTTP requests. HTTP requests are first fielded by a Web server that is configured with WebSphere Application Server. These requests are then relayed to other WebSphere Commerce Payments components.

IBM WebSphere Application Server

Why configure your Web server with WebSphere Application Server? Two reasons:

1. WebSphere Application Server is a product that works with a variety of Web servers on multiple platforms.
2. WebSphere Application Server provides an environment where the WebSphere Commerce Payments servlets can be executed.

WebSphere Commerce Payments Servlet

The WebSphere Commerce Payments Servlet, also called the Payment Servlet, is the heart of the WebSphere Commerce Payments. It receives requests from the User Interface or other merchant applications and after performing security checks, invokes a Payment Cassette to handle any necessary payment processing such as communication with financial networks, and finally sends the responses back to the calling application. It is the only component that can connect to the WebSphere Commerce Payments database and access the secure data stored on it.

Database access

Configuration and runtime information that is internal to the WebSphere Commerce Payments is stored in a relational database. WebSphere Commerce Payments is compatible with the following relational databases:

- UDB
- Oracle

To ensure that you have the correct version of your database software for use with WebSphere Commerce Payments, refer to the *WebSphere Commerce Payments Installation Guide*.

Note: You should never access the database directly. WebSphere Commerce Payments uses the database to securely store your data and provides query commands that allow you to access the data objects.

The WebSphere Commerce Payments and e-commerce entities

To this point, we have examined how WebSphere Commerce Payments:

- Interacts with merchant software
- Supports multiple payment protocols

In this section, we actually put the pieces together and examine how the WebSphere Commerce Payments supports the merchant’s shopping software. We look at three scenarios in which WebSphere Commerce Payments, in conjunction with merchant software and generic payment protocols, interacts with the following e-commerce entities:

- Merchant
- Buyer
- Financial Institution

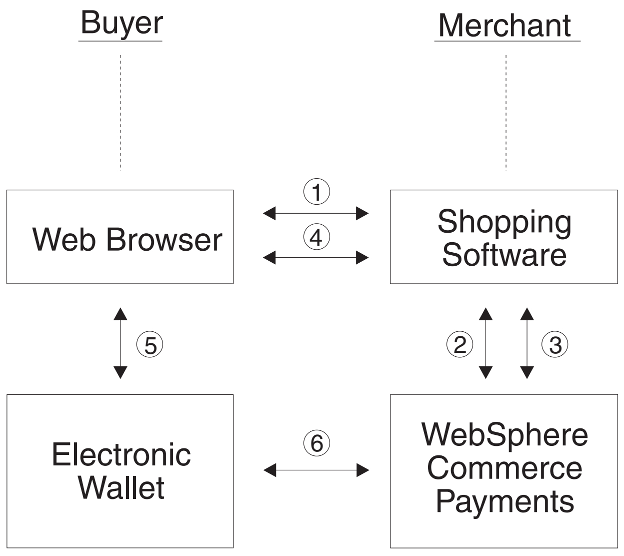
The examples that follow illustrate e-commerce using the WebSphere Commerce Payments. It is important to note that these scenarios are generic and do not represent any interactions specific to a particular payment protocol. Rather, they explore basic transaction flows common to many e-commerce business scenarios.

Scenario 1: cash and an electronic wallet

This first example uses a fictitious cash protocol and involves two entities: a Buyer and a Merchant. Funds are to be transferred from the Buyer to Merchant. In this transaction, the Buyer uses a protocol-specific electronic wallet application. The transaction steps are illustrated in the scenario diagram below as follows:

1. Interaction between the Buyer and the Merchant shopping software concludes with the Buyer issuing a purchase request to the Merchant (that is the consumer tells the Merchant that he is ready to purchase).
2. In response to the purchase request, the Merchant software sends a payment initiation command (that is, receive payment) to the WebSphere Commerce Payments.
3. The WebSphere Commerce Payments responds with a payment initiation message which contains protocol-specific information for the Buyer's wallet program.
4. The Merchant software forwards this payment initiation message to the Buyer's browser.
5. The payment initiation message received by the Buyer's browser spawns the Buyer's wallet program.
6. The Buyer completes the transaction through interactions with the WebSphere Commerce Payments and the wallet software.

Scenario 1: WebSphere Commerce Payments interacting with cash and an electronic wallet:



What separates this example from the two that follow is that it demonstrates an e-commerce transaction involving only two entities:

- The Buyer (represented by the electronic wallet)
- The Merchant (using the WebSphere Commerce Payments)

No Financial Institution (for example, a bank) is involved.

Scenario 2: credit card and an electronic wallet

The second example illustrates a purchase using a fictitious credit card protocol and involves three entities:

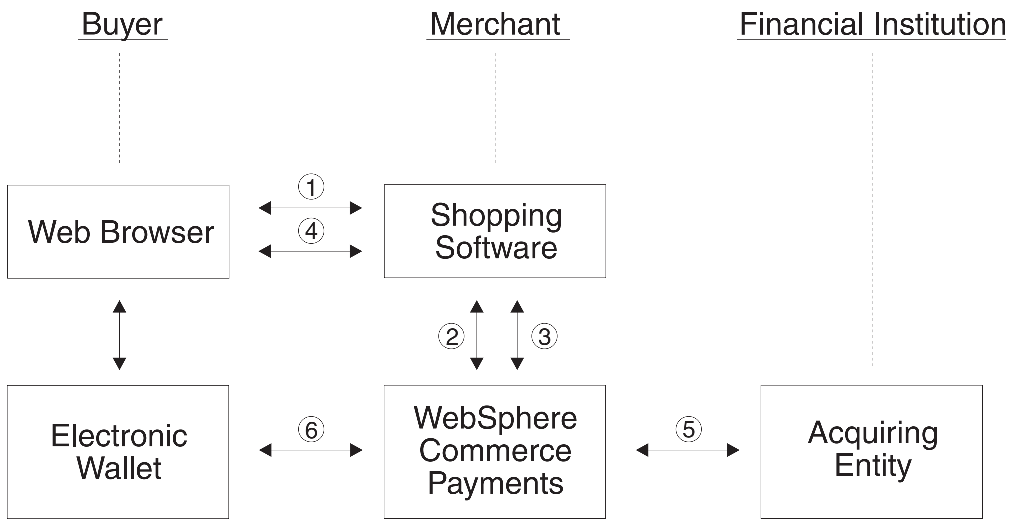
- Merchant
- Buyer (again using an electronic wallet)
- Financial Institution

This example flows similarly to scenario 1 (that is, cash and the electronic wallet) with the following differences:

- When the Buyer's wallet completes the purchase request, the WebSphere Commerce Payments issues an approval request to the Financial Institution's acquiring entity to ensure sufficient cardholder funds.
- The transaction concludes with the WebSphere Commerce Payments sending a confirmation response to the Buyer's wallet indicating approval status.

The scenario diagram illustrates this variance in steps five and six below:

Scenario 2: WebSphere Commerce Payments interacting with a credit card and an electronic wallet:



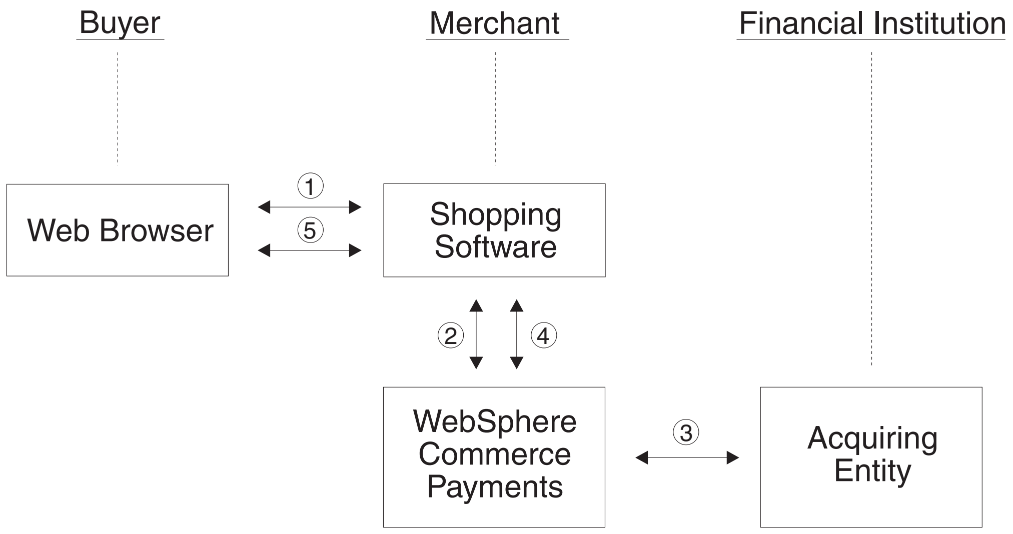
With the addition of steps five and six, the Financial Institution's acquiring entity confirms sufficient cardholder funds, while the WebSphere Commerce Payments, in turn, confirms to the Buyer that fund deposit has occurred.

Scenario 3: credit card and SSL

The final example illustrates a credit card protocol where the Buyer has no wallet, or consumer-specific, software. All consumer information is passed to the merchant software in a secure manner – such as in an SSL socket. The transaction steps are illustrated in the scenario diagram below as follows:

1. Interaction between the Buyer and the Merchant shopping software concludes with the Buyer issuing a purchase request to the Merchant (that is the consumer tells the Merchant that he is ready to purchase).
2. In response to the purchase request, the Merchant software sends a payment initiation command to the WebSphere Commerce Payments.
3. The WebSphere Commerce Payments attempts to approve the transaction (by confirming sufficient cardholder funds) through the acquiring entity's Financial Institution.
4. The WebSphere Commerce Payments returns a result (such as, fund deposit has occurred) to the Merchant software.
5. Finally, the Buyer is notified of the purchase request's success or failure.

Scenario 3: WebSphere Commerce Payments interacting with a credit card and SSL:



The preceding diagrams illustrate possible e-commerce scenarios and demonstrate the WebSphere Commerce Payments's role within those scenarios. It is important to note that many other business flows are possible, and the WebSphere Commerce Payments will assume a similar payment function in those interactions.

Chapter 2. Getting started after installation

This chapter is intended for first-time WebSphere Commerce Payments users and covers the basics of getting up and running after WebSphere Commerce Payments installation is complete. At this point, you should have completed the following for workstation platforms:

- Installed the WebSphere Commerce Payments Framework
- Started the Web server

If you are using an iSeries system, at this point you should have completed the following:

- Installed the WebSphere Commerce Payments Framework
- Created a WebSphere Commerce Payments instance
- Added a cassette to the WebSphere Commerce Payments instance
- Started the WebSphere Commerce Payments instance

If you need assistance in performing the previous steps, refer to the WebSphere Commerce Payments Installation Guide.

The following sections guide you through a first-time WebSphere Commerce Payments setup. As part of this initial setup, and to demonstrate the most common administration and payment functions, WebSphere Commerce Payments provides tutorial support using the OfflineCard cassette and a Sample Checkout. For detailed information on administration, configuration and payment functions, see the Online Help for the WebSphere Commerce Payments user interface.

Using the tutorial software as a model, this chapter demonstrates everything you *must do* to achieve a fully functioning WebSphere Commerce Payments. In addition, this chapter walks you through fictitious scenarios that simulate real-world functions. And while you need not complete the entire walk through, it is important that you step through the first six, *must-do*, tasks in order to attain an operational WebSphere Commerce Payments that is truly *ready-for-business*. Following are the six *must-do* tasks:

1. Define WebSphere Commerce Payments users
2. Start the WebSphere Commerce Payments user interface
3. Create a WebSphere Commerce Payments merchant and authorize a payment cassette
4. Assign user roles
5. Create an account
6. Create a brand

Note: In the real-world of e-commerce, the buyer shops online using the merchant's Internet storefront (that is, shopping software) and create orders via a *buy* option on the storefront.

Once the orders are created, you are ready to begin the payment processing tasks that merchants typically perform on a daily basis:

1. Approve orders
2. Deposit payments
3. Settle batches
4. Issue credits
5. View daily batch totals

It is recommended that you take the time to walk through all sections of this tutorial. If you choose not to do so, the WebSphere Commerce Payments user interface provides Online Help for fields where you require assistance.

Defining WebSphere Commerce Payments users on iSeries systems

By default, the WebSphere Commerce Payments is configured to use the Native OS User Realm which is composed of native OS/400 users. Users must have a valid OS/400 user profile prior to being assigned roles in the WebSphere Commerce Payments user configuration (through the WebSphere Commerce Payments user interface). The Create User Profile (CRTUSRPRF) CL command can be used by the iSeries security administrator to create a user profile. For purposes of this tutorial, you will work with two users:

1. QPYMADM - The default Payments Administrator
2. Pat - A user you create with the CRTUSRPRF command.

If you are installing WebSphere Commerce Payments on a system remote from WebSphere Commerce products and would like to use the WebSphere Commerce stylesheet in your WebSphere Commerce Payments user interface, you must copy the `PMCustomUI.properties` file located in the `/QIBM/ProdData/PymSvr/Samples/PMCustomUI.properties` directory to the `/QIBM/UserData/PymSvr/<pminstance>` directory (where `<pminstance>` is the WebSphere Commerce Payments instance).

Defining WebSphere Commerce Payments users on Windows and UNIX systems

For this tutorial, you will work with two users:

1. *wcsadmin*
2. *Pat*, a user which you will define

If you have one of the WebSphere commerce products installed on a remote machine from WebSphere Commerce Payments, then the steps that you follow to complete this tutorial will be somewhat different. For example, when one of the WebSphere commerce products is installed remotely, then the WebSphere Commerce Payments predefines a *default user* called `admin` with the default password of `admin`. Whereas, when a WebSphere Commerce product is installed on a the same machine, WebSphere Commerce Payments predefines the user, `wcsadmin`, as the Payments administrator in the WebSphere Commerce Payments user interface.

If WebSphere Commerce Payments detects a WebSphere Commerce product, then you will use the WebSphere Commerce administrative console to accomplish tasks such as defining and managing users. To define the user, *Pat*, you must use the WebSphere Commerce administrative console. The steps to define new users are explained in the WebSphere commerce software product Online Help.

Also, if you are installing WebSphere Commerce Payments on a system remote from WebSphere Commerce products and would like to use the WebSphere Commerce stylesheet in your WebSphere Commerce Payments user interface, you must copy the `PMCustomUI.properties` file located in the `<PMInstallDir>\samples\wcs\PMCustomUI.properties` directory path to the main WebSphere Commerce Payments installation directory.

Notes:

1. WebSphere Commerce Payments allows you to define new users (such as *Pat*) through the PSDefaultRealm when a WebSphere Commerce product is installed on a remote machine. For more information on defining users through the PSDefaultRealm, see "PSDefaultRealm" on page 34.
2. When the Merchant administrator requires additional user IDs, they must be created and assigned according to that particular realm's instructions.

Starting the WebSphere Commerce Payments user interface

Having been assigned the default user, `wcsadmin`, you are ready to start the WebSphere Commerce Payments user interface and logon as the Payments administrator.

Depending on whether you have a Windows or UNIX platform or an iSeries system, you will be working with two users in this tutorial, either `admin` and `Pat` or `QPYMADM` and `Pat`. After the Payments administrator grants you access, you can log on to the WebSphere Commerce Payments user interface for the first time as your user name using the same name for the password. With iSeries systems, however, log on to the WebSphere Commerce Payments user interface for the first time as QPYMADM, using the password assigned by your security administrator.

To access the WebSphere Commerce Payments user interface:

1. Point your browser to `http://<hostname>/webapp/PaymentManager/`, where `<hostname>` is the machine where the WebSphere Commerce Payments is installed.

   Note: If the HTTP Server that the WebSphere Commerce Payments instance is using is configured for a port number other than the default of 80, it must also be specified following the hostname in the WebSphere Commerce Payments URL links referenced throughout this section.
2. At the WebSphere Commerce Payments Logon window, type the Payments administrator user ID and password, and select **OK**.

The icons in the upper right window of the user interface have the following uses:

- Select the *multi-direction arrow* to refresh the window.
- Select the *back arrow* (that is, the left-pointing arrow) to return to the last page visited. Use the back arrow instead of your browser's back button.
- Select the *question mark* to access online Help for the window.

Setting the browser locale

It is important to note that the WebSphere Commerce Payments user interface attempts to display information according to your preferred locale (that is, a combination of language and country code). Most Web browsers allow you to specify a list of locales and send this list to Web servers. The WebSphere Commerce Payments user interface uses this list to determine the following:

- The natural language to use
- The locale's preferred number format (for example, in Germany, one thousand dollars and ninety-nine cents is shown as $1.000,99)
- The locale's preferred date format
- The locale's preferred collation order (for example, in France, the collation order specifies whether *`e* is displayed before *e*)

At your initial logon, the user interface looks at the preferred locale list and, for each of the above items, selects the best match. If no match can be found (for example, you speak Swahili), then the user interface tries to find a match for the default locale of the machine on which the user interface is installed. If all else fails, the user interface uses an en_US locale.

Creating a WebSphere Commerce Payments merchant and authorizing a cassette

You are now logged into the WebSphere Commerce Payments as the Payments administrator and so have global views and global authority. The first step in configuring the WebSphere Commerce Payments is to create a merchant and authorize that merchant to use a payment cassette:

1. From the navigation frame, click **Merchant Settings** under the Administration section.
2. From the Merchant Settings window, click **Add a Merchant**.
3. At the next window, you are prompted to enter the following (note that the italicized text *must* be entered in these fields for the tutorial):

Merchant name	Enter *Test Store*. This is the name that you assign to the merchant. Its only function is to provide display information in the user interface.
Merchant number	Enter *123456789*. This is a number that you assign which uniquely identifies the merchant in all transaction data.
Authorized cassettes	Check the box next to the *OfflineCard cassette*. Checking this box authorizes the merchant to use this payment cassette.

4. When you have entered the requested information, click **Create Merchant** to save the merchant configuration.

Assigning user roles

Having created:

- A user, *Pat*
- A merchant, *Test Store*

you are ready to assign Pat’s role in the WebSphere Commerce Payments configuration.

Users must be assigned to one of the following WebSphere Commerce Payments roles:

- Payments administrator
- Merchant administrator
- Supervisor
- Clerk

Note: Users can also be assigned *no WebSphere Commerce Payments access*. Users who are granted *no WebSphere Commerce Payments access* are not able to access the WebSphere Commerce Payments. For more information on WebSphere Commerce Payments role permissions, see the Role Permissions Table in the *WebSphere Commerce Payments Programmer’s Guide and Reference*.

For the purposes of this tutorial, you will assign Pat the role of Merchant administrator for the Test Store.

1. From the navigation frame, click **Users** under the Administration section.
2. From the Users Search window, enter the user name **Pat** and click Search.

   Note: On the User Search window, you can search the WebSphere Commerce Payments database for a specific user.
3. From the Users window, click the user name **Pat**.
4. Select the merchant **Test Store** from the available merchants in the drop-down list.
5. Select **Merchant administrator** as the user role.
6. When you have entered the requested information, click **Update** to save the user configuration.

It is important to note that in hosted environments (that is, where a Commerce Service Provider (CSP) establishes a WebSphere Commerce Payments that remotely services multiple merchants), the CSP serves as the Payments administrator, creating merchants, authorizing the merchant to use payment cassettes, and creating merchant administrators per each merchant's contract with the provider. In the hosted world, the merchant would configure his own merchant settings with Merchant administrator authority granted him by the CSP (that is, Payments administrator). This is the model we are adopting for this tutorial.

At this point, you should logoff the WebSphere Commerce Payments user interface and logon again, this time as the Merchant administrator, Pat.

Note: A merchant can use the WebSphere Commerce Payments as a non-hosted (that is, single enterprise) solution. And in a non-hosted environment, the merchant can function as *both* the Payments administrator and the Merchant administrator; in which case, the merchant could continue configuring his environment while logged in as the Payments administrator. If you will not be using the WebSphere Commerce Payments as a hosted solution, you can remain logged in as the Payments administrator (QPYMADM or admin).

Fuzzy searches can only be performed on the user name.

Note: If you are a Merchant administrator, you can search for only users that have been assigned to a merchant for which you are authorized.

Logging in as the Merchant administrator

To logoff and logon again:

1. Select **Logoff** *user id* on the navigation frame of the WebSphere Commerce Payments user interface to return to the main WebSphere Commerce Payments Logon window.
2. From the main WebSphere Commerce Payments Logon window, type the user ID `Pat` along with the password defined for Pat and select **OK**.

You are now logged in to the WebSphere Commerce Payments user interface as user Pat, with Merchant administrator authority for the Test Store merchant. For the remainder of the tutorial, you will act as the Merchant administrator. You will notice that your view of the WebSphere Commerce Payments user interface is now limited to *merchant* administration functions; whereas, as the Payments administrator, you had a global view of both *merchant* and *WebSphere Commerce Payments* administration.

Creating an account

So far, you have defined one merchant, the Test Store, and enabled one payment cassette, the OfflineCard cassette. Now, your first task as the Merchant administrator is to establish an *account* for the OfflineCard cassette.

An account is a relationship between the merchant and the financial institution which processes transactions for that merchant. There can be multiple accounts for each payment cassette. But for the purposes of this tutorial, you will create one account for the OfflineCard cassette.

To create an account:

1. Click **Merchant Settings** on the navigation frame of the WebSphere Commerce Payments user interface.
2. From the Merchant Settings window, click the OfflineCard cassette icon in the Test Store window.
3. From the OfflineCard cassette window, click **Accounts**.
4. Click **Add an Account** on the Accounts window.
5. At the next window, you will be prompted to enter the following information (note that the italicized text *must* be entered in these fields for the Sample Checkout to function properly):

Account name	Enter *Test Account*. This is the name that you assign to the account. Its only function is to provide display information in the user interface.
Account number	Enter *111111111*. This is a number that you (that is, either the hosting service provider or the merchant administrator) assign which uniquely identifies the account in all transaction data.
Financial Institution name	Enter *Test Bank*. This is the name of the financial institution with which you hold this account. Its only function is to provide display information in the user interface.
Currency	Enter *US Dollar*
Batch close time	The time of day at which the open batches are to be closed. If not specified, all batches must be explicitly closed through the BatchClose command. This value is specified as an integer representing the number of minutes past midnight (where 0 represents midnight). Valid values are 0 <= time <= 1439.

6. Click **Create account** to create an account for the OfflineCard cassette.

Creating a brand

Now that you have created the account for the OfflineCard cassette, you need to create a *brand* for the Test Account. A brand is a credit card type, such as VISA or MasterCard. The brand, or brands, that you define for an account is based upon the terms of the account as defined by the merchant.

To create a brand:

1. Click **Merchant Settings** on the navigation frame of the WebSphere Commerce Payments user interface.
2. From the Merchant Settings window, click the OfflineCard cassette icon in the Test Store window.
3. From the OfflineCard cassette window, click **Accounts**.
4. Click **Test Account** on the Accounts window.
5. From the Test Account window, click **Brands**.
6. Click **Add a brand** on the Brands window.
7. At the next window, you will be prompted to enter the following information (note that the italicized text *must* be entered in these fields for the tutorial).
8. Enter **ROBO**.
9. Click **Create brand** to create a brand for the Test Account.
10. Click **Logoff Pat** on the navigation frame.

Specifying Account Settings

To use the tutorial, it is not necessary to specify any particular basic or advanced settings for the account and brand you just created for a cassette. You should be aware, however, that you can specify certain options for the account. For example, as Merchant administrator, you can define whether the account should have an expiration period associated with payment approvals.

To view account settings from the WebSphere Commerce Payments user interface for the OfflineCard cassette, do the following:

1. Select **Merchant Settings** on the navigation frame.
2. From the Merchant Settings window, click the OfflineCard cassette icon in the Test Store window.
3. From the OfflineCard cassette window, click **Accounts**.
4. Select **Test Account** on the Accounts window.
5. From the Test Account window, select **Account Settings**.
6. You can view or change basic or advanced settings by clicking on the appropriate selection. For example, **select Advanced** settings to see the Approval Expiration setting for the Automated Payment Processing Option.
7. Enter options as appropriate and select **Update** to update the account settings. (To cancel, select the Back arrow on the Account Settings window.)

You can refer to the Online help for more information about settings for the processing options. For now, however, we will explore one of them—the Approval Expiration option.

You may want to set an approval expiration to control the length of time a payment approval is valid. It may be useful to set an approval expiration period to avoid charges imposed by your financial institution for depositing funds after an approval expires. For example, for credit card orders, payment approval lasts for only a certain number of days as established by the credit card issuer. If you try to deposit funds for an order payment after the approval has expired, your financial institution may either reject the funds deposit or charge you more to make the deposit. By setting an approval expiration period in WebSphere Commerce Payments you can avoid this situation because WebSphere Commerce Payments will prevent the capturing of funds for which an approval authorization has expired.

If a payment approval has expired and you try to deposit funds, you will receive a message indicating the payment is not in the approve state. In the WebSphere Commerce Payments user interface, you can void the payment with an amount of 0. Once the payment is voided, you can perform an Approve operation to place the payment back into the approve state.

To use the approval expiration option, the cassette you use must support approval expiration. The SET, CyberCash, OfflineCard, VisaNet and CustomOffline cassettes provided with WebSphere Commerce Payments support approval expiration.

Note: The value you enter as the approval expiration period applies to all brands associated with an account. If your brands each have different payment approval expiration rules, you should set the approval expiration value for the lowest common denominator, or create a separate account for each brand. If you create a separate account for each brand, you will need to settle batch payments separately for each account.

More information about approving orders, depositing payments, and settling batches is provided later in this chapter.

Managing payment processing

As the Merchant administrator, you have global *merchant* authority, which means that you can perform:

- Merchant-specific administration functions
- Payment processing functions (all)

In a real business scenario, you may choose to delegate payment processing tasks to other merchant-defined users who possess limited payment processing authorities (such as, supervisor and clerk). In this tutorial, you, as the Merchant administrator, will perform these tasks.

Having completed all of the WebSphere Commerce Payments and Merchant administration tasks necessary to begin payment processing, you are now ready to start:

- Approving orders
- Depositing payments
- Settling batches
- Issuing credits
- Viewing daily batch totals

For the purposes of this tutorial, you will use the SampleCheckout to create three orders for use in payment processing.

Creating orders using the Sample Checkout

As previously discussed, a real business environment features a buyer who creates orders using a merchant's Internet storefront and a merchant who processes payments for those orders using the WebSphere Commerce Payments. In order for you to walk through the WebSphere Commerce Payments payment processing functions, you need to create orders that require payment processing. To simulate a merchant's Internet storefront, and thus facilitate order creation, the WebSphere Commerce Payments supplies a sample tool called the Sample Checkout. To access the WebSphere Commerce Payments Sample Checkout and create orders:

1. Go to the WebSphere Commerce Payments directory `Install/samples/SampleCheckout` subdirectory or `QIBM/UserData/PymSvr/<instance>`, where `instance` is the name of the WebSphere Commerce Payments instance. Edit the configuration file called `SampleCheckout.xml`. This file specifies the user ID and password that are to be used to authenticate the Payment Manger user in the <SampleCheckout> element.

   When you install WebSphere Commerce Payments with a WebSphere Commerce product, WebSphere Commerce Payments automatically enables WCSRealm if the WebSphere Commerce product is installed on the same machine as WebSphere Commerce Payments. The default Payments Administrator user ID is `wcsadmin`. Thus, if the WCSRealm is enabled, the password attributes for the <SampleCheckout> element in the SampleCheckout.xml file should be manually updated to reflect the correct user ID and password.

   Notes:

   a. If WebSphere Commerce Payments is using the WCSRealm to authenticate users, your WebSphere Commerce product must be running.
   b. If you are using an iSeries system, verify that the host name and port number in the <SampleCheckout> element are set properly for your WebSphere Commerce Payments instance.
2. Make sure the following line is present under the <PaymentOptionList>:

   ```
   <PaymentOption id="OfflineCard" profile="SampleCheckoutOfflineCard">
   OfflineCard</PaymentOption>
   ```
3. Point your browser to **http://<hostname>/webapp/PaymentManager/SampleCheckout**, where `<hostname>` is the machine where the WebSphere Commerce Payments is installed.
4. At the Sample Checkout window that appears, you will be prompted to enter the following (note that the italicized text *must* be entered in these fields for the tutorial):

Merchant number	Enter *123456789*
Order number	Any number that hasn't been previously used in the tutorial to represent an order number
Amount	Enter *10.00*
Currency	Enter *US Dollar*
Payment method	Choose *OfflineCard* to process the payment with the OfflineCard cassette
Brand	Enter *ROBO*
Credit card number	Enter *4111111111111111*.
Expiration date	Select the expiration month and year for your ROBO credit card. Note that you can choose any month and year combination for this tutorial.

5. Select **Buy**.

Repeat these steps two additional times so that you have three orders for which to process payments, changing the order number for each new order.

Approving orders

Once you have created three orders using the Sample Checkout, return to the browser window where the WebSphere Commerce Payments user interface is displayed. Point your browser once again to the WebSphere Commerce Payments URL (that is, `http://<<hostname>>/webapp/PaymentManager/`) and logon as `Pat`. Follow these steps to approve an order:

1. From the navigation frame, click **Approve** under the Payment Processing section.
2. From the Approve window, check the box next to the order you want to approve (select only one order for this exercise) and click **Approve Selected**.
3. At the Approve Results window, you will see the status of your approve request. When processing is complete, success or failure status will appear next to each order submitted for approval. When your approval is complete, click **Return to the Approve screen**.

Two orders are still awaiting your approval. You could have approved them all at once (for their full amount) by clicking **Approve All** from the Approve window. But to better demonstrate the many facets of the Approve function, you will work with each order individually.

Approving orders from the Order window

In this section, you will approve an order from the Order window (rather than from the Approve window), but you will approve only *part* of the total order amount. You may find it useful to approve only part of an order when some of the goods associated with the order are not available for delivery at order processing (for example, merchandise that is on back-order).

1. From the Approve window, click the **Order number** for one of the remaining orders awaiting approval.
2. From the Order window, you can view order details. Click **Approve** to approve this order.
3. From the Order Approve window, the following information displays:

Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	This field displays zeros since no amount of the order has yet been approved. This is a read-only field.
Deposited Amount	This field displays zeros since no amount has yet been approved or deposited. This is a read-only field.
Approval Amount	This is the total amount of the order.
Authorization Code	The authorization code returned from the manual offline authorization request process. The payment state is changed to ″Approved″.
Decline Reason	The decline reason returned from the manual offline authorization process. The payment state is changed to ″Declined″.
AVS Result Code	The AVS result code returned from the manual offline authorization request process.

Change the approval amount to `3.00` and click **Approve** to approve this order for three dollars.

When approval processing has completed, you will be returned to the Order window and notified of approval success or failure. You will notice in the order details that the approved amount has been updated to reflect the three dollars we specified in the previous step.

Using the Sale function to approve orders

Because you approved only *part* of the last order you worked with, you still have two order entries in the Approve window. In this exercise, you will use the *sale* function to approve the remaining orders.

The sale function allows you to approve an order and move it directly into Deposited state, bypassing Approved state. The sale function automatically performs an Approve and a Deposit on your order payment (thus, you can think of sale as Approve with auto-deposit). Use the sale function when you want to expedite the delivery of goods to the buyer and guarantee your capture of funds (for example, when you are selling downloadable software or electronic information). However, with the Sale function, you lose the ability to set Authorization reason or Decline reason on the approval. Because sale provides you with the mechanism to merge Approve and Deposit into one transaction, the sale function is also useful when you are charged on a per-transaction basis. Perform a sale as follows:

1. From the navigation frame, click **Approve** under the Payment Processing section.
2. Click **Sale All** from the Approve window.
3. At the Approve Results window, you will see a progress bar indicating the status of your sale request. When processing is complete, success or failure status will appear next to each order submitted for sale.
4. When your sale is complete, click **Return to the Approve Screen**.

Depositing payments

Deposit allows you to deposit order payments. As demonstrated in "Approving orders from the Order window" on page 20, a single order number can have multiple payments associated with it. You may see the same order number appear multiple times in the same list, each time with different payment information. To deposit a payment:

1. From the navigation frame, click **Deposit** under the Payment Processing section.
2. Check the box next to one of the payments listed and click **Deposit Selected**.
3. When processing is complete, success or failure status will appear in the Deposit Results window next to the payment submitted for deposit.
4. When your sale is complete, click **Return to the Deposit Screen**.

Note: You can deposit only *part* of a payment, in much the same way you approved part of an order:

1. From the Deposit window, click the **Payment number** for one of the payments awaiting deposit.
2. The next window is the Payment window. From the Payment window, you can view payment details. Click **Deposit** to deposit this payment.
3. On the Order Payment window, change the deposit amount to `2.00` and click **Deposit** to deposit this payment for two dollars.

Settling batches

A batch is a collection of payments and credits that are processed as a unit by a financial institution. A batch is associated with a merchant and an account. The payments that you deposited in the previous exercise will now appear in a batch. You must *settle* this batch in order to initiate processing by the financial institution. To settle a batch:

1. From the navigation frame, click **Batch Search** under the Payment Processing section.
2. At the Batch Search window, you will be prompted to enter the following information (note that for the purposes of this tutorial, you will not be entering any parameter information in the fields to narrow your search) :

Merchant	The name of the merchant whose batch you are searching for. **Note:** If there are less than 500 merchants in the WebSphere Commerce Payments database, you select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, you enter the name of a merchant.
Batch Number	The number that uniquely identifies the batch within the merchant. Assigned when the payment is deposited.
State	The state of the batch: • Open • Closed
Balance Status	The balance status of this batch: • **Balanced:** the batch has been successfully balanced (that is, all totals agree). • **Out of balance:** an unsuccessful attempt has been made to balance this batch (that is, all totals do not agree).
Payment Type	Identifies the payment type, or protocol, used to place the order (for example, OfflineCard).
Batch Open Date	Use the *after* and *before* fields below to search for batches opened during the specified range in time: • **After**: Specify a date to search for all batches opened on and after this date. • **Before**: Specify a date to search for all batches opened on and before this date.
Batch Closed Date	Use the *before* and *after* fields below to search for batches closed during the specified range in time: • **After**: Specify a date to search for all batches closed on and after this date. • **Before**: Specify a date to search for all batches closed on and before this date.

3. Click **Search** to initiate a batch search.

   Note: In addition to using the *after* and *before* fields to specify a time range for the batch search (such as, 08/01/2001 to 08/15/2001). These fields can also be used to narrow search results by *excluding* certain batches from the search. For example, you could search on all batches opened *before* 08/01/2001 and all batches opened *after* 08/15/2001, thus excluding batches opened between 08/02/2001 and 08/14/2001.
4. Click the batch number to view information about the batch.

5. From the Batch window, you can view useful batch information, including the total number and amount of both payments and credits in the batch. Click **Batch Details** to see a detailed listing of all payments and credits in this batch.
6. Click **Settle** to settle the batch.
7. When processing is complete, success or failure status will appear in the Settle Results window.

When the success message displays, the settle command is complete. The financial institution is now responsible for the transfer of funds.

Note: You also have the option to delete the settled batch by clicking **Delete** on the Settle Results window. When a batch is deleted, all ancillary information about that batch (that is, payments, credits, and cassette-specific data) is deleted, as well. If you need to retain all payment data (for example, for audit purposes), then you should not delete a batch. But if you need to prune outdated information, you can exercise the **Delete** Batch option.

Issuing a credit

Credits are issued against orders and can be given for any amount. To issue a credit, you need to find the order against which you are issuing the credit:

1. From the navigation frame, click **Order Search** under the Payment Processing section.
2. On the Order Search window, you will be prompted to enter the following information (note that for the purposes of this tutorial, you will not be entering any parameter information in the fields to narrow your search):

Merchant	The name of the merchant whose order you are searching for. **Note:** If there are less than 500 merchants in the WebSphere Commerce Payments database, select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, enter the name of a merchant.
Order Number	A number assigned by the merchant that uniquely identifies the order.
State	The state of the order: • Requested • Ordered • Refundable • Canceled • Closed
Payment Type	Identifies the payment type, or protocol, used to place the order (for example, OfflineCard).
Order Date	Use the *after* and *before* fields below to search for orders opened during the specified range in time: • **After**: Specify a date to search for all orders opened on and after this date. • **Before**: Specify a date to search for all orders opened on and before this date.

Order Amount	• **Currency**: The currency used to place this order. Select the currency type from the drop-down list. • **Greater than**: Specify a value to retrieve all orders with order amounts that are greater than or equal to the value you specify. • **Less than**: Specify a value to retrieve all orders with order amounts that are less than or equal to the value you specify.

3. Click **Search** to initiate an order search.

   Note: In addition to using the *after* and *before* fields to specify a time range for the order search (such as, 08/01/2001 to 08/15/2001). These fields can also be used to narrow search results by *excluding* certain orders from the search. For example, you could search on all orders opened *before* 08/01/2001 and all orders opened *after* 08/15/2001, thus excluding orders opened between 08/02/2001 and 08/14/2001.
4. From the next window, click an order number for an order in Refundable state to view the details of that order.
5. From the Order window, click **Credit** to create a credit against this order.
6. At the Create Credit window, the following information displays:

Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	The total amount of the order that has been approved expressed in the currency used to place the order. This is a read-only field.
Deposited Amount	The total amount of the order that has been deposited expressed in the currency used to place the order. This is a read-only field.
Credit Amount	This field must be completed by the merchant administrator with the total amount to be credited to the shopper.
Authorization Code	The authorization code returned from the manual offline authorization request process. The credit state is changed to ″Refunded″.
Decline Reason	The decline reason returned from the manual offline authorization process. The credit state is changed to ″Declined″.

   Enter the credit amount (can be *any* amount) and click **Credit**.

When credit processing has completed, you will be returned to the Order window and notified of credit success or failure. You will notice on the Order window that the newly created credit appears under **Credits** at the bottom of the window.

Viewing batch totals

The last exercise in this tutorial is viewing daily batch totals. The WebSphere Commerce Payments Reports function allows you to view *daily totals* for batches in a closed state. To generate a daily batch totals report:

1. From the navigation frame, click **Reports** under the Payment Processing section.
2. From the Reports window, click **Daily Batch Totals**.

3. At the Batch Totals Report window, you will be prompted to enter the date for which you would like a batch totals report. WebSphere Commerce Payments computes the batch totals for the date entered and generates a report. *Leave this field blank to generate a report for the current date*.
4. You will also be prompted to enter the Merchant name. If you do not enter a merchant name, a list of all of the batches for the specified date will display. If there are more than 500 batches, the first 500 batches will display and you will be prompted to narrow your search by selecting a specific merchant.
5. Click **Search** to initiate a batch report search.

The Daily Batch Totals report computes the totals for all batches that were closed on the date specified on the **Search** window. Since you did not specify a search date, the report that was generated contains the current day's batch totals. These totals are computed on a per-currency basis, so there is one line per currency. Note that these totals cover all payments and credits made for *all payment types* (not just those made through the OfflineCard cassette).

You have just completed a day in the life of a Payments administrator and a Merchant administrator. While individual business models may vary, this tutorial outlines the basic path to establishing a working WebSphere Commerce Payments and demonstrates fundamental payment processing implemented through the WebSphere Commerce Payments. For more information on specific fields in the WebSphere Commerce Payments user interface, see the Online Help.

Chapter 3. Maintaining WebSphere Commerce Payments security

WebSphere Commerce Payments security is built on several key security elements. These elements combine to create an environment in which secure Web services can be deployed.

This chapter is intended for security administrators or anyone who is responsible for managing WebSphere Commerce Payments security.

Protecting the WebSphere Commerce Payments

As discussed in “WebSphere Commerce Payments: internal components” on page 5, the heart of WebSphere Commerce Payments is the Payment Servlet. Several ancillary products, the Web server configured with WebSphere Application Server, the database, and the user interface, complete the WebSphere Commerce Payments picture. This chapter discusses methods for securing the various WebSphere Commerce Payments components.

User Guidelines for iSeries systems

Only Payments administrators that manage WebSphere Commerce Payments instances, such as creating instances or adding cassettes, need access to the WebSphere Commerce Payments commands and the WebSphere Commerce Payments Tasks page. A user can be authorized to manage instances by specifying QPYMSVR as a group profile using the Change User Profile (CHGUSRPRF) CL command.

The WebSphere Commerce Payments instances run under the user profile QPYMSVR, which is created when the product is installed.

Additionally, users that need to administer or perform payment processing for a WebSphere Commerce Payments instance will need access to the WebSphere Commerce Payments User Interface. The default Payments administrator for all WebSphere Commerce Payments instances is the QPYMADM user profile. The QPYMADM user profile has access to perform all administration and payment processing tasks for all WebSphere Commerce Payments instances through the WebSphere Commerce Payments User Interface, but does not have access to the WebSphere Commerce Payments Tasks page, the CL commands or the database tables.

To assign additional unique Payments administrators for each instance, log onto the WebSphere Commerce Payments User Interface for that instance using the QPYMADM user profile and configure additional users to access the WebSphere Commerce Payments. See “Assigning user roles” on page 14 for more information.

WebSphere Commerce Payments password

The WebSphere Commerce Payments instance password is required when managing a WebSphere Commerce Payments instance. If the system security officer has set the QRETSVRSEC system value to allow server security data to be retrieved from a validation list, the WebSphere Commerce Payments instance password will be automatically retrieved as needed by the WebSphere Commerce Payments. Otherwise, you will be required to enter the password when managing an instance.

Protecting sensitive data

A property within the PSDefaultRealm.properties file specifies the minimum role a user must have in order to view sensitive data. For each query command, the framework verifies the user's role against that minimum role and thereby, sets an indicator in the QueryRequest object to indicate whether sensitive data should be returned in full view or if it should be masked out. Currently, the WebSphere Commerce Payments framework does not maintain any sensitive data that can be returned via a query command. However, new methods are provided to cassette writers to check the value of this indicator and also to mask sensitive data in a standardized way. Each cassette must discern the sensitive data from the rest of the stored data. Typically, the sensitive data is the same set of data that a cassette encrypts before storing it to the WebSphere Commerce Payments database.

The `wpm.MinSensitiveAccessRole={clerk|supervisor|admin|psadmin|none}` property specifies the minimum role a user must have to be allowed access to sensitive data. The value is case-sensitive. If this property is not specified, a value of clerk is assumed, allowing all users to see sensitive data. If an invalid value is specified, the Payment Servlet fails to initialize. The following table describes supported values, which are listed in increasing order of authority:

clerk	Users with a role of clerk or higher can see sensitive data.
supervisor	Users with a role of supervisor or higher can see sensitive data.
madmin	Users with a role of merchant administrator or higher can see sensitive data.
psadmin	Only Payments administrators can see sensitive data.
none	No one is allowed to see sensitive data.

You can specify the `wpm.MinSensitiveAccessRole` parameter through the WebSphere Application Server administrative console. Complete the following to modify the values for this parameter:

1. Open the WebSphere Application Server Administrative console.
2. Expand **WebSphere Administrative Domain**.
3. Expand **Nodes**.
4. Expand the hostname for the system where WebSphere Commerce Payments is installed.
5. Click **WebSphere Commerce Payments**. If you are using an iSeries system, click **WPM** *<instance>* **WebSphere Commerce Payments**, where *<instance>* is the name of the WebSphere Commerce Payments instance.
6. Select the **JVM Settings** tab page. In the **System Properties** box, add the parameter name `wpm.MinSensitiveAccessRole`.
7. Add the default role of `clerk` as the parameter value.
8. Click **Apply**.
9. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

Protecting the database on Windows and UNIX systems

The WebSphere Commerce Payments database stores sensitive information and requires protection from reading and writing by unauthorized sources. The WebSphere Commerce Payments provides support for the encryption of sensitive data – for example, passwords and cardholder information – that is stored in the database.

Changing the WebSphere Commerce Payments database password

On Windows and UNIX systems, WebSphere Commerce Payments secures sensitive data by using the password of the database user ID that you use to connect to the WebSphere Commerce Payments database. Therefore, when changing the database password, you must identify the new and old password to WebSphere Commerce Payments so that it can re-secure your data.

WebSphere Commerce Payments does not require that you follow the approach described below to change your database password. However, once you have changed the database password, you must use the `IBMPayServer -changepassword` command to re-encrypt your sensitive data. Use your database tools or other operating system tools to perform the following steps to change the WebSphere Commerce Payments database password:

1. Stop the WebSphere Commerce Payments using the WebSphere Application Server Administrative console. Highlight the **WebSphere Commerce Payments Application Server** object and select the **Stop** button.
2. If you use the same user ID and password to connect to the WebSphere Commerce Payments database as you do for any other applications (for example, your Web server or WebSphere Application Server), then you should also stop those applications and make sure the password change will be correctly reflected in those systems.

   Note: For security reasons, it is not recommended that you use the WebSphere Commerce Payments user ID to connect to anything other than WebSphere Commerce Payments.
3. Change the database password using whatever system tool with which you are familiar.
4. Start the WebSphere Commerce Payments Application Server using the WebSphere Application Server Administrative console. Highlight the **WebSphere Commerce Payments Application Server** object and select the **Start** button.
5. Initialize WebSphere Commerce Payments using the **IBMPayServer** script and the **-changepassword** option, specifying the old and new passwords:

   ```
   IBMPayServer -changepassword
   ```

   Executing this command will prompt you to enter both your old database password and your new one. WebSphere Commerce Payments will use these passwords to re-secure your sensitive data.

   Note: If you use the **-file** option of the `IBMPayServer` command for unattended operation, then your password file will be updated with the new database password as long as you have write access to that password file. Refer to the *WebSphere Commerce Payments Installation Guide* for more information on unattended operating systems.

Attention

After your WebSphere Commerce Payments database has been created, you should not modify the payment database tables in any way. Table modification can cause the payment system to function incorrectly, or possibly fail.

Protecting the database on iSeries systems

The database stores sensitive information and requires protection from reading and writing by unauthorized sources. WebSphere Commerce Payments provides support for the encryption of sensitive data—for example, passwords and cardholder information—that is stored in WebSphere Commerce Payments.

Transaction data

- Sensitive transactional information is stored in a database table in the instance library. This library is created by the Create WebSphere Commerce Payments (CRTPYMMGR) CL command with a public authority of *EXCLUDE. You should ensure that only Payments administrators have access to this library.
- Any backups should be kept secure.
- The database tables in the library instance contain critical configuration and transaction information and should be included as part of your system backup strategy. You should also back up the following:
  - Files in the /QIBM/UserData/PymSvr/Instance/ directory where Instance is the name of the WebSphere Commerce Payments instance
  - HTTP server instance that you configured for WebSphere Commerce Payments
  - Objects in the instance library on the local machine, as well as, the database collection on the remote machine when remote database storage is used
  - Any files in the /QIBM/UserData/PymSvr/ directory

Note: In some circumstances, the WebSphere Commerce Payments instance may not appear in the list of available instances after being restored on a system. If this occurs, you can use the Start WebSphere Commerce Payments (STRPYMMGR) CL command to start the instance. After the WebSphere Commerce Payments is started, the instance will be displayed in the list of available instances.

Changing the WebSphere Commerce Payments password

The WebSphere Commerce Payments uses a password to encrypt sensitive information such as its encryption keys. The WebSphere Commerce Payments password can be changed by selecting the Change Password menu item from the WebSphere Commerce Payments Tasks page. When the password is changed, the sensitive information will be re-encrypted with the new password.

Attention: If you lose your password, there is no way to recover encrypted information or certificates. This is different from other iSeries system passwords where the security officer can reset the password for you and nothing is lost. It is recommended that you back up your data just prior to changing the password.

Changing the WebSphere Commerce Payments Database Password

The database password used to connect to the remote database for a WebSphere Commerce Payments instance will need to be changed if the password is changed on the remote machine. The database password can be changed from the Database menu item from the WebSphere Commerce Payments Tasks page.

Attention

After your WebSphere Commerce Payments database has been created, you should not modify the payment database tables in any way. Table modification can cause the payment system to function incorrectly, or possibly fail.

Protecting the Web server using SSL

In order to protect the information exchanged between the remote merchant's Web browser and the WebSphere Commerce Payments Web server, you must secure the communication channel. Secure Socket Layer (SSL) is a general-purpose network security protocol that can be used to provide the secure communication between a client and a Web server. The `https` protocol is the combination of SSL and `http` protocol. Security features provided by SSL are:

- **Server authentication:** A Web server is authenticated by providing its digital certificate to the client; the client, in turn, authenticates the digital certificate based on a trusted Certificate Authority.
- **Client authentication:** Optionally, the client provides its digital certificate to the server which, in turn, then authenticates the client based on a trusted Certificate Authority.
- **Confidentiality:** All client requests and server responses are encrypted to maintain the confidentiality of the data exchanged over the network.
- **Data integrity:** Data that flows between a client and a server is protected from a third party's tampering.

All of the supported WebSphere Commerce Payments Web servers support SSL, and it is recommended that you enable your Web server for SSL. You can install a server certificate on these Web servers for them to present to a client during an `https` request. They can be optionally configured to authenticate a client, based on the requested resource. Each of the Web servers has its own interface to manage keys, key rings, server certificates, and client certificates. As described earlier, after SSL is installed and enabled on a server, all communication between the server and SSL-enabled browsers are private, authenticated, and checked for message integrity. For more information on certificate management and on installing and enabling SSL, refer to the respective Web server documentation.

Setting up IBM HTTP Server for SSL on Windows and UNIX systems

If you choose to install WebSphere Commerce Payments's default Web server, IBM HTTP Server, the server will be set up as non-secure during the installation program, and the installation program will update the httpd.conf file accordingly. If you want SSL support, you must perform additional configuration steps. See the documentation provided with the IBM HTTP Server for instructions on configuring the HTTP Server for SSL.

Protecting the user interface using SSL

There are two WebSphere Commerce Payments communication channels that should be protected:

1. The link between the user's Web browser and the WebSphere Commerce Payments user interface

2. The link between the WebSphere Commerce Payments user interface and the Payment Servlet

It is recommended that you use SSL to secure both communication links.

SSL between the browser and the user interface

When the WebSphere Commerce Payments is initially installed, SSL protection is disabled. Perform the following to enable SSL protection:

1. Follow your Web server documentation to obtain SSL certificates. Note that the Web server on the computer where you installed the user interface must support SSL.
2. Start the user interface by pointing your browser to either of the following URL addresses:
   - `https://<<hostname>>/webapp/PaymentManager`
   - `http://<<hostname>>/webapp/PaymentManager`

   Note: `<<hostname>>` is the hostname of the machine where you installed the WebSphere Commerce Payments user interface.

   By pointing your browser to the `https` protocol (the first URL address in the list), the WebSphere Commerce Payments user interface will be protected by SSL; by pointing your browser to the `http` protocol (the second URL address in the list), the WebSphere Commerce Payments user interface will *not* be protected by SSL.
3. On Windows and UNIX platforms, the WebSphere Commerce Payments is installed with a file called `index.html` in the WebSphere Commerce Payments subdirectory of your Web server's publish directory. On iSeries systems, the `index.html` file is installed in the /QIBM/ProdData/HTTP/Protect/PymSvr/UI directory. If you want to ensure that the WebSphere Commerce Payments user interface is always protected by SSL, even when users point their browsers to `http://<<hostname>>/webapp/PaymentManager`, you can do so by using the supplied `indexSSL.html`:
   a. Copy `indexSSL.html` to `index.html`.
   b. Then, edit `index.html` to make sure that the hostname specified inside the file is the same as the hostname specified in your SSL certificate.

      Note: You can disable SSL protection by overwriting `index.html` with `indexNonSSL.html`.

SSL between the user interface and the Payment Servlet

It is important to note that securing communication between the WebSphere Commerce Payments user interface and the Payment Servlet is necessary only if the user interface is installed on a *different* machine other than the WebSphere Commerce Payments itself.

Note: The Client API Library (CAL) is used to connect from the user interface servlet to the Payment Servlet. In WebSphere Commerce Payments, Version 3.1, CAL trusts *all* SSL certificates presented to it by the Web server it connects to. No SSL certificate management tool is provided with CAL. This is not a security issue if both servlets are on the same machine, but security may be compromised if the user interface servlet is moved to a different machine. The iSeries version of CAL relies on SSL support within the Developer Kit for Java, which authenticates server certificates and can provide a client certificate to the server if a system default certificate has been defined on the system. For more information on CAL, see the *WebSphere Commerce Payments Programmer's Guide and Reference*.

You can use the WebSphere Application Server administrative console to secure communication between the WebSphere Commerce Payments user interface and the Payment Servlet. To secure this communication, complete the following to modify the values for this parameter:

1. Open the WebSphere Application Server Administrative console.
2. Expand **WebSphere Administrative Domain**.
3. Expand **Nodes**.
4. Expand the hostname for the system where WebSphere Commerce Payments is installed.
5. Click **WebSphere Commerce Payments**. If you are using an iSeries system, click **WPM** *<instance>* **WebSphere Commerce Payments**, where *<instance>* is the name of the WebSphere Commerce Payments instance.
6. Select the **JVM Settings** tab page. In the **System Properties** box, enter the SSL parameter value.
7. Click **Apply**.
8. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

When modifying these parameters through the WebSphere Application Server console, ensure that:

- The `wpmui.PaymentServerPort` property is set to the SSL port of the Web server.
- The default SSL port number is 443.
- The `wpmui.DisableSSL` property is set to 0.

WebSphere Commerce Payments access

WebSphere Commerce Payments authenticates users through the use of realms. A realm is a registry of users along with a single method of authenticating those users (for example, a user’s name and password). Each WebSphere Commerce Payments installation can only use one realm at a time. Examples of realm types include LDAP realms and operating system realms. A user must be defined in a realm before being granted access to resources. Therefore, a user is a valid WebSphere Commerce Payments user if, *and only if,* he is both:

- in the realm
- assigned a role in the WebSphere Commerce Payments

WebSphere Commerce Payments employs a role-based access control scheme which defines four WebSphere Commerce Payments roles:

1. Payments administrator
2. merchant administrator
3. supervisor
4. clerk

The Payments administrator can use the WebSphere Commerce Payments user interface User window to assign access (based on role) to a user defined in a realm. Though other realms can be created, the following are provided with the WebSphere Commerce Payments:

- **PSDefaultRealm:** This is a simple realm implementation provided by the WebSphere Commerce Payments which uses a flat file to hold a list of realm users and their corresponding passwords. You may implement this realm when you are using WebSphere Commerce Payments with version 4.0 of the WebSphere Application Server.

- **PSOS400Realm**: This is the default realm for WebSphere Commerce Payments running on iSeries systems. The WebSphere Commerce Payments validates users and their corresponding passwords against existing OS/400 user profiles. Users are added and deleted through OS/400 commands.
- **WCSRealm**: The WCSRealm class is automatically configured for your system if WebSphere Commerce Payments is installed on the same machine as WebSphere Commerce Suite. This realm allows the WebSphere Commerce Payments Servlet to use the administrator information that is already registered in the WebSphere Commerce Suite user tables. This administrator information is used for Payments administrators so that you do not have to define another set of administrator IDs in order to use the WebSphere Commerce Payments user interface.

As previously noted, these are not the only mechanisms for authenticating and authorizing users. Other applications that use the WebSphere Commerce Payments may override our realm implementation and use their own support. For information on writing new realms for use with the WebSphere Commerce Payments, see the *WebSphere Commerce Payments Programmer's Guide and Reference*.

PSDefaultRealm

For users with an installation of WebSphere Application Server, Version 4.0 on a remote machine, WebSphere Commerce Payments provides realm support in the form of the **PSDefaultRealm**. The PSDefaultRealm is a simple realm implementation that uses flat file to hold a list of realm users and their corresponding passwords. Each entry in this file is Base64 encoded to provide a basic level of security.

Adding and deleting users in the PSDefaultRealm

You can add or delete users in the PSDefaultRealm using the WebSphere Commerce Payments realm utility that manages the Base64 encoded realm flat file. The WebSphere Commerce Payments realm utility can be invoked from the **Payments** root directory using either:
- `PSDefaultRealm.cmd` file (on Windows)
- `PSDefaultRealm` file (on Solaris and AIX)

To add a user to the PSDefaultRealm:
1. Go to the **Payments** root directory.
2. Enter the following:
   ```
   PSDefaultRealm <<RealmFile>> add <<userName>> <<password>>
   ```
   where
   - `<<RealmFile>>` is the absolute path of the realm flat file.
   - `<<userName>>` is the name of the user you are adding to the PSDefaultRealm.
   - `<<password>>` is the password of the user you are adding to the PSDefaultRealm.

To delete a user from the PSDefaultRealm:
1. Go to the **Payments** root directory.
2. Enter the following:
   ```
   PSDefaultRealm <<RealmFile>> remove <<userName>>
   ```
   where
   - `<<RealmFile>>` is the absolute path of the realm flat file.

- `<<userName>>` is the name of the user you are deleting from the PSDefaultRealm.

Changing user passwords in the PSDefaultRealm: To change a user password in the PSDefaultRealm, you must delete the user and then add the same user with a new password. See the above instructions for deleting and adding users.

Viewing users in the PSDefaultRealm

In addition to adding and deleting users in the PSDefaultRealm, you may also need to view users who are defined in the realm. To view a list of users defined for the PSDefaultRealm:

1. Go to the **Payments** root directory.
2. Enter the following:

   ```
   PSDefaultRealm <<RealmFile>> list
   ```

   where
   - `<<RealmFile>>` is the absolute path of the realm flat file.

The `list` command will display the users defined in the PSDefaultRealm.

Creating new realm files

On workstation platforms, the WebSphere Commerce Payments default realm file is **PSRealm**. To use a `RealmFile` other than the default, you must:

- Create a new realm using the PSDefaultRealm utility.
- Change the `RealmFile` setting in `PSDefaultRealm.properties` to reflect the new realm.

You must ensure that the user `admin` (or any other user currently defined to the WebSphere Commerce Payments with Payments administrator authority) is defined in the new `RealmFile` prior to changing the `RealmFile` setting in `PSDefaultRealm.properties`. You can then use this user ID to logon to WebSphere Commerce Payments and assign user roles to other users in the new realm.

Note: To invoke the newly created realm, you need to switch from the realm currently employed by the WebSphere Commerce Payments to the new realm. For information on switching realms, see “Switching from one WebSphere Commerce Payments realm to another” on page 36.

Protecting PSRealm files

After the WebSphere Commerce Payments installation, you should protect the PSRealm files as follows:

- On Solaris and AIX, you can use the UNIX permission bits.
- On Windows systems, you need to keep the machine and the file system secured in order to protect this file.

WCSRealm

When you install WebSphere Commerce Payments on a system where a WebSphere commerce product is installed, the WCSRealm class is automatically configured for your system. If you are using WebSphere Commerce Payments from a remote machine, then the PSDefaultRealm will be provided as the supporting default realm. For more information on the PSDefaultRealm, see “PSDefaultRealm” on page 34. However, to use the WCSRealm on a remote machine where WebSphere Commerce Payments is installed you must manually configure

WebSphere Commerce Payments through the WebSphere Application Server administrative console. To manually configure your system to use the WCSRealm, complete the following:

1. Open the WebSphere Application Server Administrative console.
2. Expand **WebSphere Administrative Domain**.
3. Expand **Nodes**.
4. Expand the host name for the system where WebSphere Commerce Payments is installed.
5. Click **WebSphere WebSphere Commerce Payments**. If you are using an iSeries system, click **WPM** *<instance>* **WebSphere Commerce Payments**, where *<instance>* is the name of the WebSphere Commerce Payments instance.
6. Select the **JVM Settings** tab page. In the **System Properties** box, add the realm name, `WCSRealm`.
7. Add the parameter value **com.ibm.commerce.payment.realm.WCSRealm**.
8. Click **Apply**.
9. Edit the WCSRealm.properties file from your WebSphere Commerce Payments installation directory to change the host name:

   For the `WCSHostName` enter the fully-qualified host name of the remote machine where the WebSphere commerce product is installed. (The default is the host name of the system where WebSphere Commerce Payments is installed.) You do not have to change the property for the `WCSWebPath`.
10. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

Managing users in the PSOS400Realm

To manage users in the PSOS400Realm, you must use OS/400 user profile commands.

Table 2. User profile commands

Function	OS/400 command
Add a user	Create User Profile (CRTUSRPRF)
Delete a user	Delete User Profile (DLTUSRPRF)
Change user password	Change User Profile (CHGUSRPRF)
Viewing users	Work with User Profiles (WRKUSRPRF)

Switching from one WebSphere Commerce Payments realm to another

Because each WebSphere Commerce Payments installation can use only one realm at a time, it is useful to understand how to switch between realms. The `PSDefaultRealm.properties` file contains settings used by the Payment Servlet. These settings include the specification of which realm should be used together with any settings required by the realm itself.

To specify which realm should be used:

- Change the `RealmClass` setting in the `PSDefaultRealm.properties` file to reflect the realm you want to deploy.

After changing the `RealmClass` setting in the `PSDefaultRealm.properties` file, you must add the realm-specific properties for the realm you just specified.

You can also switch between realms by using the WebSphere Application Server Application Assembly Tool. For instructions on how to switch from one realm to another by changing the `RealmClass` property settings see the instructions in the "WCSRealm" on page 35.

Restarting your Web server and WebSphere Application Server

After updating the `RealmFile` setting and the realm-specific properties, you must restart your Web server and WebSphere Application Server for these changes to take effect. For instructions on stopping and starting your Web server and WebSphere Application Server, see the *WebSphere Commerce Payments Installation Guide*.

PSOS400Realm properties

The PSOS400Realm requires no additional properties. The following is an example of the PSDefaultRealm.properties setting needed to configure the PSOS400Realm:

```
RealmClass=com.ibm.etill.framework.payserverapi.PSOS400Realm
```

PSDefaultRealm properties

The PSDefaultRealm requires an additional `RealmFile` property to be present in the `PSDefaultRealm.properties` file. This property value should be the absolute path name of the Base64 encoded flat file. The following is an example of the `PSDefaultRealm.properties` setting needed to configure the PSDefaultRealm:

```
RealmFile=c:/myPSRealm
```

Note: The slashes in the previous example must be forward slashes – rather than *back slashes* – even on Windows operating system.

Firewall considerations

The WebSphere Commerce Payments can be used with firewall software. The extra setup steps you need to perform depend on the component with which the WebSphere Commerce Payments is trying to communicate. If you need to initiate communications outside the firewall, you must use a socks–enabled TCP/IP stack on the WebSphere Commerce Payments machine itself to act as a proxy for data flowing to and from WebSphere Commerce Payments through the firewall. For AIX and Solaris, the TCP/IP stack is already socks-enabled. For Windows systems, you must obtain a third-party product. Examples are Hummingbird's socks-enabled TCP/IP stack and Aventail's AutoSOCKS.

For an iSeries system, firewall considerations depend on the component with which the WebSphere Commerce Payments is trying to communicate. For example:

- If the merchant system is outside the firewall, you must configure a firewall filter to allow a limited set of merchant IP addresses/ports to access WebSphere Commerce Payments's address/payment API port. Likewise, you configure a firewall filter to permit any wallet address/port to access WebSphere Commerce Payments SET address/payment port and address/inquiry port.
- If the payment gateway is outside of a socks-enabled firewall, you must configure the iSeries socks client support using Operations Navigator to allow WebSphere Commerce Payments to communicate with the Payment Gateway.

General security practices

In addition to the recommendations already noted in this chapter, consider the following points:

- Run anti-virus software regularly, particularly on Windows platforms. One approach is to use the IBM AntiVirus products.
- A real-time intrusion detection system should be deployed to monitor access to the network where the WebSphere Commerce Payments product is installed.
- Consider physical security measures for the room or area where the WebSphere Commerce Payments is running.
- Use security file monitoring software.
- Read industry publications dealing with firewalls and security on the Internet. Examples are:
  – Garfinkel, Simson, and Gene Spafford. *Practical UNIX and Internet Security.* Cambridge, MA: O'Reilly and Associates, 1996. (ISBN: 1565921488)
  – Garfinkel, Simson, and Gene Spafford. *Web Security and Commerce.* Sebastopol, CA: O'Reilly and Associates, 1997. (ISBN: 1565922697)
  – Rubin, Aviel D., et al.*Web Security Sourcebook.* New York: Wiley Computer Pub., 1997. (ISBN: 047118148X)

Chapter 4. Customization of WebSphere Commerce Payments

Customizing your user interface

You can customize the look and feel of the WebSphere Commerce Payments user interface to suit yourself. This activity is called *branding*. Branding can be universal across all languages, or you can choose to limit changes to a single language. Branding is performed in two ways:

1. By creating Java properties files. These allow you to modify the text for the options and actions presented on the screen.
2. By changing the WebSphere Commerce Payments style sheet or creating additional style sheets. The style sheets control background color, text and table specifications, and other screen attributes.

Branding with Java properties files

WebSphere Commerce Payments does not initially contain an installed PMCustomUI properties file. When you create a properties file, its contents override the WebSphere Commerce Payments default values. There are three ways you can use properties files:

1. Create a global properties file. The contents of this file will override the WebSphere Commerce Payments defaults for all languages.
2. Create a unique properties file for each of the available languages. In this case, you can make changes for those languages that you choose, and leave the files for the other languages blank. Note that you *must* create a file for every language, even if that language is not used or if the properties for that language are not to be modified. These properties files override the WebSphere Commerce Payments defaults.
3. Create both a global properties file *and a complete set* of properties files for each of the languages. In this case, the language-specific properties file takes priority. If it is blank, the global properties file overrides the WebSphere Commerce Payments defaults.

Your Java properties files may be located wherever you wish, but they must be pointed to in the classpath that is set through WebSphere. These files must be in unicode.

Note: When you install WebSphere Commerce Payments on a remote machine, you must locate the PMCustomUI.properties file in the**<Payments_installdir\samples\wcs>** directory and manually place it in the **<Payments_installdir>** directory

How to find the objects to be modified

When you have identified an item that you want to modify, open WebSphere Commerce Payments and go to the WebSphere Commerce Payments screen where the object is found. Then use your browser to display the html source for that page by right-clicking on the frame and choosing **View source**. (Note that some browsers will not display the source with this method.) The information you need is an identifier that is associated with the object in the html. Each comment begins with ″PMTID″, followed by the type of the object, then the object’s identifier. In the following example, the object is a Field with the identifier: payment.orderSearch.orderNumber.merchantNumber. Note that there are no spaces and the parts of the identifier are separated by periods.

```
<!– PMTID FIELD=payment.orderSearch.orderNumber.merchantNumber –>
```

Creating your properties files

Using an editor, place one line in your properties file for each object to be changed. The line begins with the object identifier, followed by an equal sign (=) and then the new value to be applied to that object. Here are some examples of actual lines in a properties file. The last word before the equal sign is always an option or action and is always in capital letters.

```
admin.HEADER=This is the new header of the admin component
reports.TRAILER=Here is a new trailer for the reports
payment.orderSearch.state.state.order_requested.BUTTON=Hot button
```

To remove the tool bar that contains the Help, Back and Refresh icons, insert the following message in the custom properties file: **admin.ISHOWHELP.MESSAGE=0**.

The highest level of object is *component*. There are three *components* in WebSphere Commerce Payments; admin, payment, and reports. Therefore, there is no single line that can be written in a properties file to make a change throughout all of WebSphere Commerce Payments. You must write at least one line per component. The list of items that can be changed for a component are:

Name	appears in the title bar of the browser
Header	appears at the top of every screen in the component, often contains an image
Trailer	appears at the bottom of every screen in the component, may include an image or text

An example of changing all headers within a component is:

```
payment.HEADER=new_header
```

The next level of object is the *screen*. The list of items that can be changed for a screen are:

Name	appears at the top of the screen under the component's header
Header	appears under the screen's name, only displayed on screens that have a default header
Trailer	appears at the bottom, just above the component's trailer, only displayed on screens that have a default trailer
emptylist	appears when a WebSphere Commerce Payments query is made with no results found from the search

An example of changing the name of a screen is:

```
payment.screen1.NAME=new_name
```

The list of items that can be changed for a fieldgroup are:

Name	appears centered on the banner above the fieldgroup name
Header	appears left-aligned above the fields of a fieldgroup
Trailer	appears left-aligned below the fields of a fieldgroup

An example of changing the trailer for all the fields that use trailers in a particular fieldgroup is:

```
payment.screen1.fieldgroup3.TRAILER=new_trailer
```

The next level is *field*. The list of items that can be changed for a field are the following:

Name	appears to the left of the field in a colored box
Shorthelp	appears to the right of the field to provide help information for the field content
Defaultvalue	appears in the text box as the default text
Preferredwidth	width of the field defined as whole number of pixels or percent
Preferredheight	height of the field defined as whole number of pixels or percent
Maximumlength	sets maximum number of characters for the field
ColumnPosition	indicates where a column field appears
Fieldposition	indicates where a field appears in a detailed screen view
Columnwidth	width of the column defined as whole number of pixels or percent

An example of changing the columnwidth of a field called ″somename″ to 15% of screen width is:

```
payment.screen1.fieldgroup3.somename.COLUMNWIDTH=15%
```

Without the percent sign, the columnwidth would be set to 15 pixels.

To change the text associated with an action, such as a push-button, use the following as an example:

```
payment.screen.ACTION=Next
```

In this example, the text on the button will be changed to ″Next″. You can do the same with radio buttons.

Naming your properties files

If you create a global properties file, it must be named **PMCustomUI.properties**. Properties files for specific languages must be named **PMCustomUI_*xx*.properties** where the *xx* represents the language locale.

Remember, to make changes for individual languages, you must create a complete set of files for all the languages. To make changes to the French version only of your WebSphere Commerce Payments, you must put your changes in the **PMCustomUI_fr.properties** and leave the files for the other languages blank.

Your Java properties files may be located wherever you wish, but they must be pointed to the classpath that is set through WebSphere. It is recommended that you place the files in the following directories since these are already configured in the WebSphere Application Server classpath:

- WebSphere Commerce Payments installation directory (for Windows, Sun OS, Linux and AIX systems)
- /QIBM/UserData/PymSvr/instance_name directory (for iSeries systems)

In addition, it is important that you convert all of your PMCustomUI.properties files to ASCII–encoded Unicode to prevent unexpected run-time character conversion. To accomplish this you can use the **native2ascii** program that is shipped with the Java Development Kit. For example: `native2ascii -encoding <orig_encoding> PMCustomUI_xx.properties.native PMCustomUI_xx.properties` where `xx` is the locale of the properties files you are converting (See the current language locales in the table that follows.) and `<native_encoding>` is the encoding of the input file **PMCustomUI_xx.properties.native**.

Following is the current list of languages and their assigned character values:

Table 3. Current language locales

Brazilian Portuguese	**pt**
Dutch	**nl**
English	**en**
French	**fr**
German	**de**
Italian	**it**
Japanese	**ja**
Korean	**ko**
Simplified Chinese	**zh**
Spanish	**es**
Traditional Chinese	**zh_TW**

Note: The Dutch (nl) language system and associated characters are not included on iSeries platforms.

Replacing images

To replace the header image for all screens of a component, determine the component name by looking at the html source for one of the screens that you want to change, and in your properties file include the following:

```
<component_name>.HEADER=<image>
```

<image> is replaced by the html reference to an image, like this:

```
<IMG SRC="/images/newimage.gif" width="584" \
height="75" BORDER="0" ALT="New Image''s ALT text">
```

To replace a header image for a single screen do this:

```
<component_name>.<screen_name>.HEADER=<image>
```

Note: The backslash (\) is a line continuation character. If a single quote is needed within double quotes, you must use two single quotes.

Branding your WebSphere Commerce Payments style sheet

The style sheet defines the background color, text specifications (size, font, color, underlining, boldness), table specifications (borders, colors, text) and other attributes. WebSphere Commerce Payments defaults to a single, cascading style sheet for the entire user interface.

The default style sheet is located at **<installDir>/style.css** on Windows and UNIX systems, where **<installDir>** is the WebSphere Commerce Payments installation directory. On iSeries systems, the default style sheet is located in the **/QIBM/ProdData/HTTP/Protect/PymSvr/UI/style.css** path. Rather than editing this file, you can copy it, make your changes to the copy and then point to the copy in your properties file. This protects you from overwriting your changes when you apply service.

To use a particular style sheet, use the identifier *styleSheet* in your properties file, followed by the location and name of the new style sheet. For example, if you create a new style sheet at /web/newStyleSheet.css, then the line in your properties file would be:

```
styleSheet=/webapp/PaymentManager/newStyleSheet.css
```

Note: If you upgrade or reinstall WebSphere Commerce Payments, you it's a good idea to have your branding properties files and style sheets backed up.

Customizing the WebSphere Commerce Payments user interface

The User Interface Servlet, called the *UI Servlet* is a Web-browser based application that allows you to both manage payments and administrate WebSphere Commerce Payments. The User Interface is like any other merchant application in that it communicates with the WebSphere Commerce Payments Servlet to perform all of its processing. The User Interface is also customizable to the extent that you can change the text, colors and style of any User Interface screen. This ability allows you to adapt WebSphere Commerce Payments to a more seamless integration with other applications in your systems.

If you are the provider of a hosted WebSphere Commerce Payments, you may want to customize the WebSphere Commerce Payments user interface to ensure that all of your merchants have a fixed set of payment method names from which to select during account creation. This provides consistency across all merchants and helps avoid the situation where some merchants support 'C.O.D.' and others support 'COD'. If you are running a WebSphere Commerce Payments as a single merchant, then you may skip this step. The CustomOffline Cassette is used in the examples shown in this section.

After you decided which methods to use for manually collecting payments, you can update the WebSphere Commerce Payments user interface to seamlessly support your chosen methods. After the user interface is updated, the payment methods are selectable when an account is being created for the CustomOffline cassette.

To update the user interface with your payment method names, update the `PMCustomUI.properties` file. The `PMCustomUI.properties` file is the properties file which must be modified to do branding (customization) of the WebSphere Commerce Payments user interface. This file is not provided during the WebSphere Commerce Payments installation, so you will need to create it if you have not already done so. Any changes which you make to `PMCustomUI.properties` will appear in all languages supported by the user interface. See Chapter 4, “Customization of WebSphere Commerce Payments” on page 39 for more information.

If you wish to provide language-specific changes, you will need to create (or edit your existing) `PMCustomUI_`*`xx`*`.properties` files, where *xx* is represents a supported language. [See “Naming your properties files” on page 41 for the list of supported options.] For example, if you want ″Cash On Delivery″ to appear when the user interface is viewed in English, edit PMCustomUI_en.properties.

For each payment method that you want to support, you make an entry in the properties file. The entry will specify the text to display in the user interface and the value which will be passed for the ″$METHOD″ parameter of the CreateAccount API call. The format of these entries is as follows:

```
CustomOffline.CustomOfflineAccount.CustomOfflineAccountDetails.Method.index=
apiValue:displayValue
```

- apiValue is the value used on the CreateAccount API call. This value must correspond to the $METHOD value used for AcceptPayment commands for this payment method. The apiValue must not contain any colons.
- displayValue is the value displayed in the user interface.
- index is a unique index of the method name used to order the method names in the user interface. The index value for your first payment method should be 1 and other entries should be numbered consecutively.

Note: Entries are case-sensitive and the apiValue and displayValue must be separated by a colon. For example:

```
CustomOffline.CustomOfflineAccount.CustomOfflineAccountDetails.Method.1
=COD:Cash On Delivery
CustomOffline.CustomOfflineAccount.CustomOfflineAccountDetails.Method.2
=BillMe:Bill me later
```

You can also customize your user interface to view the **Transaction Identifier** and **Order Data 1** fields. Both fields allow you to specify information that helps to identify one or more WebSphere Commerce Payments Orders. For example, you can specify this field to appear in the **Order**, **Order Search**, **Approve** and **Deposit** windows. In order to specify where in your user interface the fields will appear, you can add the following sample entries to your PMCustomUI.properties file:

```
payment.PSOrder.order.transactionID.FIELDPOSITION=3
payment.PSOrder.order.orderData1.FIELDPOSITION=4

payment.approve.approve.transactionID.COLPOSITION=3
payment.approve.approve.orderData1.COLPOSITION=4

payment.deposit.deposit.transactionID.COLPOSITION=3
payment.deposit.deposit.orderData1.COLPOSITION=4

payment.orderSearch.orderNumber.transactionID.FIELDPOSITION=2
payment.orderSearch.orderNumber.orderData1.FIELDPOSITION=3

payment.orderSearchResults.results.transactionID.COLPOSITION=3
payment.orderSearchResults.results.orderData1.COLPOSITION=4
```

Note: The previously mentioned COLPOSITION and FIELDPOSITION property values are meant as examples only.

After you have made your changes to the properties file, you must restart the WebSphere Application Server for the changes to take effect. For more information on these fields, refer to the WebSphere Commerce Payments Online Help.

Appendix A. OfflineCard cassette supplement

OfflineCard cassette is a *passive* cassette. Passive cassettes record events that have already happened outside of the WebSphere Commerce Payments, but within the WebSphere Commerce Payments object model. With OfflineCard cassette, transactions are recorded and maintained only in the WebSphere Commerce Payments database. There is no back-end financial system with which a passive cassette communicates. For example, an Approve command through the OfflineCard cassette records the results of an approval request which the merchant made through his/her existing credit card swipe box.

By including this cassette, WebSphere Commerce Payments is usable, right out of the box.

Using the OfflineCard cassette, the merchant can:

- Collect credit/debit card information in the WebSphere Commerce Payments through his online store using a standard *buy* page. This information is entered into the WebSphere Commerce Payments using the AcceptPayment command, which stores the data in a new Order object.
- Retrieve credit/debit card information through the WebSphere Commerce Payments user interface for use in manual credit card operations (for example, entering the credit card information through an existing swipe box).
- Record the state of the manual credit card transactions in the WebSphere Commerce Payments. For example, once a purchase is approved through the merchant's swipe box, the merchant would use the Approve button on the user interface Order window to record the fact that the approval was received. Note that transaction recording can be performed for all payment, credit, and batch-oriented operations.
- Manage daily batches using the WebSphere Commerce Payments user interface batch and reports functions.
- Perform all of these operations using the same interfaces as the merchant uses for online transactions.

Getting Started

For workstation platforms, you should have completed the following at this point:

- Installed and configured the WebSphere Commerce Payments Framework
- Started the Web server
- Started WebSphere Application Server
- Started WebSphere Commerce Payments Application Server using the WebSphere administrative console
- Installed and configured the WebSphere Commerce Payments Framework
- Initialized WebSphere Commerce Payments from the IBMPayServer command file in the install directory
- Created a merchant and Merchant Administrator for that merchant

If you are using an iSeries system, at this point you should have completed the following:

- Installed the WebSphere Commerce Payments framework
- Created a WebSphere Commerce Payments instance

- Added a cassette to the WebSphere Commerce Payments instance on the iSeries system
- Started the WebSphere Commerce Payments instance (which automatically starts the Web server, WebSphere Application Server and the Payment Servlet)
- Defined a WebSphere Commerce Payments user with administrative authority
- Created a merchant and Merchant administrator for that merchant

To configure a cassette, you must logon to WebSphere Commerce Payments as a Merchant administrator. For more information, see the *WebSphere Commerce Payments Install Guide*. In addition, Chapter 2, “Getting started after installation” on page 11 provides you with a complete tutorial for starting and using the WebSphere Commerce Payments with the OfflineCard cassette.

Overview

With the OfflineCard cassette:

- Transactions are recorded only in the local WebSphere Commerce Payments database. There is no ″back-end″ financial system with which the cassette must communicate.
- The cassette is installed with the WebSphere Commerce Payments Framework only. There is no separate install/uninstall program for this cassette.
- All batches are opened implicitly. The BatchOpen command is not supported.
- Each open batch represents a currency; therefore for each account, there can be only one open batch.
- The ReceivePayment command is not supported. All Orders are created using the AcceptPayment command.
- The AcceptPayment command can be used to create an independent credit. Merchants can use the independent credit to issue a customer refund.

Installing and configuring the OfflineCard cassette

The WebSphere Commerce Payments must be installed before the CustomOffline cassette can be installed. Installing WebSphere Commerce Payments first ensures that all prerequisite products are available for the cassette. The minimum framework that this cassette supports is WebSphere Commerce Payments Version 3.1.

Creating orders using the Sample Checkout

To access the OfflineCard cassette Sample Checkout and create orders:

1. Point your browser to http://<hostname>/webapp/PaymentManager/SampleCheckout/, where <hostname> is the machine where the WebSphere Commerce Payments is installed.
2. At the Sample Checkout window, you will be prompted to enter the following. (Note that italicized text must be entered in these fields for the tutorial.)

Merchant Number	Enter any number to represent a Merchant number
Order Number	Enter any number to represent an Order number
Amount	Enter any amount to represent the total numeric amount of the order

Currency	Enter *US dollar*. The currency used to place this order.
Payment Option	Choose *OfflineCard* as the payment option (that is, credit card) that you are using
Credit card number	Enter *4111111111111111*
Expiry Date	Highlight the expiration month and year for your credit card. (You can choose any future month and year.)

Note: If you are using an iSeries system, you must specify the Brand of the credit card that you are using.

3. Select **Buy**.

Repeat these steps two additional times so that you have three orders for which to process payments.

OfflineCard cassette example

The general operation of the OfflineCard cassette in a typical merchant scenario is as follows:

- When an online shopper buys an item through the merchant's online store, a buy page is presented to collect the shopper's credit card information. Once the shopper submits this information through the buy page, the merchant online store software submits an AcceptPayment command to the WebSphere Commerce Payments. Since the payment method chosen is *OfflineCard*, the OfflineCard cassette is called to process the protocol-specific information (that is, PAN, Expiry, and Brand) and record it along with the new Order object in the WebSphere Commerce Payments database. The shopper's payment information is now recorded so that the merchant can use it later.
- At the end of the day, the merchant uses the WebSphere Commerce Payments user interface to find all of the online orders received for the day through the OfflineCard payment method. For each of these, the merchant enters the credit card information through an existing swipe box, receives an approval or decline code, and records each of these in the WebSphere Commerce Payments using the Approve button on the Order window (which issues an Approve command). Note that an entry field is provided so that the corresponding approval/decline code can be recorded.
- Assuming the purchase is approved, the merchant packages the goods that the shopper ordered, ships them, and then records the fact that they now deserve payment by using the Deposit button on the associated Payment window (which issues a Deposit command). This places the payment in the currently open batch. If no batch is currently open, the OfflineCard cassette opens one implicitly. The OfflineCard cassette allows multiple Payments to be created for a single order, but it ensures that the combined approval amounts cannot exceed the Order amount.

  Note: When the merchant issues credits for goods bought online, these can also be recorded in the WebSphere Commerce Payments using the Refund button on the Order window (which issues a Refund command). The OfflineCard cassette allows any number of credits per order as long the total credit does not exceed the total amount of the payment associated with the order.

As mentioned above, Batch objects are created implicitly as needed when Deposit or Refund commands are processed. When the merchant is finished processing the day's payments and credits, the daily batch or batches are settled, either through the swipe box or on a timed basis by the financial institution. To record this event, the merchant can either explicitly mark the batch as having been closed using the Settle button on the user interface Batch window (which issues a BatchClose command) or by using the BatchCloseTime configuration parameter which the cassette supplies on its Account objects.

Command reference

For each WebSphere Commerce Payments API command, the following sections describe:

- How (or whether) each optional Framework parameter is supported by the OfflineCard cassette
- Any special notes related to the OfflineCard cassette's handling of Framework parameters
- All OfflineCard-specific protocol parameters

Note: For required Framework parameters, refer to the Framework objects tables.

In each table, the *Required/Optional* column has a value of **R** for required or **O** for optional.

Financial commands

AcceptPayment

The administrator can use the AcceptPayment command to create an independent credit that the merchant can use to issue a refund to the customer. For example, a customer desires a refund on an order, and the payment for that order has not been made. If the merchant decides to grant the customer a refund, the merchant can use the same order that the customer placed to issue a credit or create a new order. This allows the merchant to issue a refund on the order even when no payment exists.

Table 4. Keywords for AcceptPayment command

Keyword	**Required / Optional**	**Value**
APPROVEFLAG	O	Fully supported. ASCII integer 0, 1, or 2
PAYMENTAMOUNT	O/R	Fully supported. Required if APPROVEFLAG is set to 1 or 2. Positive integer in ASCII characters from 1 to 4,228,250,625.
DEPOSITFLAG	O	Fully supported. ASCII Integer 0 or 1
$PAN	R	Payment card number (Personal Account Number) ASCII from 12 to 19 bytes
$BRAND	R	Payment card brand. ASCII from 1 to 40 bytes
$EXPIRY	R	Payment card expiration date. ASCII 6 bytes in *yyyymm* format.
$AVS.COUNTRYCODE	O	Country code of cardholder's billing address. This field allows national language characters including DBCS. 3 to 50 bytes.
$AVS.CITY	O	City of cardholder's billing address. This field allows national language characters including DBCS. 1 to 50 bytes.

Table 4. Keywords for AcceptPayment command (continued)

$AVS.STREETADDRESS	O	Street address of cardholder's billing address. This field allows national language characters including DBCS. 1 to 128 bytes.
$AVS.STATEPROVINCE	O	City of cardholder's billing address. This field allows national language characters including DBCS, 1 to 50 bytes.
$AVS.POSTALCODE	O	Postal (zip) code of cardholder's billing address. This field allows national language characters including DBCS, 1 to 14 bytes.
$CARDHOLDERNAME	O	Cardholder name. This field allows national language characters including DBCS. 1 to 64 bytes.
$AUTHCODE	O	Authorization code to be saved with the approved Payment. This parameter is only required if APPROVEFLAG=1or 2. It is also specified on the AcceptPayment command. Must be an ASCII character string from 1 to 64 bytes long. If $AUTHCODE is specified, then a Payment is created in PAYMENT_APPROVED state for the Order amount and the ReferenceNumber is set to the $AUTHCODE value. If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_ AND_DECLINEREASON. If neither this parameter nor $DECLINEREASON is specified with APPROVEFLAG=1 or 2, then an approved Payment is created for the Order amount with an empty ReferenceNumber.
$DECLINEREASON	O	Decline code to be saved with the declined Payment. This parameter is only used if APPROVEFLAG=1or 2. It is also specified on the AcceptPayment command. Must be an ASCII character string from 1 to 254 bytes long. If $DECLINEREASON is specified, then a Payment is also created in PAYMENT_DECLINED state and the ReferenceNumber is set to the $DECLINEREASON value. If specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_ AND_DECLINEREASON
$AVSCODE	O	AVS code to be saved with the Payment. This parameter is only used if APPROVEFLAG=1or 2. It is also specified on the AcceptPayment command. Must be an ASCII character string from 1 to 3 bytes long.

Approve

Table 5. Keywords for Approve command

Keyword	Required / Optional	Value
DEPOSITFLAG	O	Fully supported.
$AUTHCODE	O	Authorization code to be saved with the approved Payment. Must be an ASCII character string from 1 to 16 characters long. If $AUTHCODE is specified, then the Payment is created in PAYMENT_APPROVED state and the ReferenceNumber is set to the $AUTHCODE value. If neither this parameter nor $DECLINEREASON is specified, then the Payment will be created in PAYMENT_APPROVED state with an empty ReferenceNumber. —If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON
$DECLINEREASON	O	Decline code to be saved with the declined Payment. Must be an ASCII character string from 1 to 254 characters long. If $DECLINEREASON is specified, then the Payment is created in PAYMENT_DECLINED state and the declineReason field is set to the $DECLINEREASON value. If not specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON
$AVSCODE	O	AVS code to be saved with the Payment. Must be an ASCII character string from 1 to 3 characters long.

BatchClose

Table 6. Keywords for BatchClose command

Keyword	Required / Optional	Value
$FIBATCHID	O	Specifies the financial institution's identifier for the batch. This parameters allows the merchant to correlate the local batch object to an entity at the financial institution. If specified, the value is saved as the fiBatchId value in the OfflineCard cassette's extension to the Batch object. Otherwise, that field is left empty. Must be an ASCII character string from 1 to 32 characters long.

Refund

Table 7. Keywords for Refund command

Keyword	Required / Optional	Value
$AUTHCODE	O	Authorization code to be saved with the approved Credit. Must be an ASCII character string from 1 to 64 characters long. If $AUTHCODE is specified, then the Credit is created in CREDIT_APPROVED state and the ReferenceNumber is set to the $AUTHCODE value. If neither this parameter nor $DECLINEREASON is specified, then the Credit is created in CREDIT_APPROVED state with an empty ReferenceNumber. If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON
$DECLINEREASON	O	Decline code to be saved with the declined Credit. Must be an ASCII character string from 1 to 254 characters long. If $DECLINEREASON is specified, then the Credit is created in CREDIT_DECLINED state and the declineReason field is set to the $DECLINEREASON value. If specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON

Other supported financial commands

The following financial commands are supported without any optional or protocol data parameters:

- **ApproveReversal** modifies the approved amount of a payment.
- **BatchPurge** clears out a batch and returns the Batch object to the Open state.
- **CancelOrder** moves an order into the cancelled state.
- **CloseOrder** moves an order into the closed state.
- **DepositReversal** disassociates a payment from a batch.
- **DeleteBatch** removes the specified batch from the database tables.
- **RefundReversal** voids existing credit objects.

Unsupported financial commands

The following financial commands are not supported and always return

```
PRC_COMMAND_NOT_SUPPORTED, RC_NONE
```

- **BatchOpen**
- **ReceivePayment**

Administration commands

CreateAccount

Table 8. Keywords for CreateAccount command

Keyword	Required / Optional	Value
$BATCHCLOSETIME	O	This parameter specifies the time of day at which the cassette should automatically close the open batches. If not specified, all batches must be explicitly closed through the BatchClose command. This value is specified as an integer representing the number of minutes past midnight (where 0 represents midnight). Valid values are 0 <= time <= 1439, null to disable function
$CURRENCY	R	ISO 3166 currency code associated with this account. A 3 digit string ranging from 001 to 999.

CreateMerchantCassetteObject

Table 9. Keywords for CreateMerchantCassette command

Keyword	Required / Optional	Value
ACCOUNTNUMBER	R	Specifies the account number. ASCII integer from 1 to 999999999.
OBJECTNAME=BRAND	R	Tells the cassette that this command is creating or deleting one of its Brand objects. ASCII string ″BRAND.″
$BRAND	R	Specifies the brand identifier. Each brand identifier must be unique within the Account. ASCII value from 1 to 40 English characters in length (case sensitive - exactly as merchants are expected to enter it).

DeleteMerchantCassetteObject

Table 10. Keywords for DeleteMerchantCassette command

Keyword	Required / Optional	Value
OBJECTNAME=BRAND	R	Tells the cassette that this command is creating or deleting one of its Brand objects. ASCII string ″BRAND.″
$BRAND	R	Specifies the brand identifier. Each brand identifier must be unique within the Account. From 1 to 40 ASCII characters long (case sensitive - exactly as merchants are expected to enter it).

ModifyAccount

Table 11. Keywords for ModifyAccount command

Keyword	Required / Optional	Value
$BATCHCLOSETIME	R	This parameter specifies the time of day at which the cassette should automatically close the open batches. If not specified, all batches must be explicitly closed through the BatchClose command. This value is specified as an integer representing the number of minutes past midnight (where 0 represents midnight). Valid values are 0 <= time <= 1439

Other supported administration commands

The following administration commands are supported without any optional or protocol data parameters:

- **CreatePaySystem**
- **DeleteAccount**
- **DeletePaySystem**
- **ModifyCassette**
- **ModifyPaySystem**

Unsupported administration commands

The following administration commands are not supported and always return

`PRC_COMMAND_NOT_SUPPORTED, RC_NONE`

- **CassetteControl**
- **CreateSystemCassetteObject**
- **DeleteSystemCassetteObject**
- **ModifyMerchantCassetteObject**
- **ModifySystemCassetteObject**

Object reference

The OfflineCard cassette object model closely reflects the generic model of the WebSphere Commerce Payments. This section describes both the financial and the administration objects and their cassette extensions to the various Framework objects.

Financial objects

The OfflineCard financial object model mirrors that of the generic model in that an OfflineCard object is defined to augment each generic financial object. The OfflineCard objects appear as extensions to the generic objects. Those extensions are as follows:

Order

Table 12. Order object cassette properties

Field name	Syntax	Description
brand	Character string, 1 to 40 bytes long	Credit/Debit card brand as specified on the ACCEPTPAYMENT command. This value is always present.
pan	Character string, 1 to 19 bytes long	Credit/Debit card number as specified on the ACCEPTPAYMENT command. This value is always present.
expiry	Character string, 6 bytes long	Credit/Debit card's expiration date in the form yyyymm as specified on the ACCEPTPAYMENT command. This value is always present.
avsCountry	Character string, 3 to 50 bytes long	The cardholder's country of residence. This value is only present if a non-null $AVS.COUNTRY value was specified on the ACCEPTPAYMENT command.
avsStreet	Character string, 1 to 128 bytes long	Cardholder's street address. This value is only present if a non-null $AVS.STREETADDRESS value was specified on the ACCEPTPAYMENT command.
avsCity	Character string, 1 to 50 bytes long	Cardholder's city of residence. This value is only present if a non-null $AVS.CITY value was specified on the ACCEPTPAYMENT command.
avsState	Character string, 1 to 50 bytes long	Cardholder's state or province of residence. This value only present if a non-null $AVS.STATEPROVINCE value was specified on the ACCEPTPAYMENT command.
avsPostalCode	Character string, 1 to 14 ASCII bytes long	Cardholder's postal code. This value is only present if a non-null $AVS.POSTALCODE value was specified on the ACCEPTPAYMENT command.
cardholderName	Character string, 1 to 64 bytes long	Cardholder's name. This value is only present if a non-null $CARDHOLDERNAME value was specified on the ACCEPTPAYMENT command.

Payment

Table 13. Payment object cassette properties

Field name	Syntax	Description
referenceNumber (framework attribute)	Character string, 1 to 16 bytes long	The authorization code entered through the $AUTHCODE parameter of the Approve command. This field is present only when an authCode value exists for this approved payment.
avsCode	Character string, 1 to 3 bytes long	The AVS code entered through the $AVSCODE parameter of the Approve command. This field is present only when an AVS code exists for this payment.
declineReason	Character string, 1 to 254 bytes long	The decline reason entered through the $DECLINEREASON parameter of the Approve command. This field is present only when a declineCode exists for this declined payment.

Credit

Table 14. Credit object cassette properties

Field name	Syntax	Description
authCode	Character string, 1 to 16 bytes long	The authorization code entered through the $AUTHCODE parameter of the Refund command. This field is present only when an authCode value exists for this credited credit.
declineReason	Character string, 1 to 254 bytes long.	The decline reason entered through the $DECLINEREASON parameter of the Refund command. This field is present only when a declineCode exists for this void refund.

Batch

Table 15. Batch object cassette properties

Field name	Syntax	Description
FIBatchID	Character string, 1 to 32 characters long	The financial institution's identifier for the batch as entered through the $FIBATCHID parameter on the BatchClose command. The merchant should use this identifier when contacting the financial institution with inquiries concerning this batch.

Administration objects

The OfflineCard extends one Framework administrative object (the AccountAdmin object) and also defines one class of MerchantCassetteObject (Brand). Multiple Accounts are supported for each merchant, and each Brand object must be associated with an Account.

Account

The OfflineCard Account extensions appear as extensions to the generic objects. Those extensions are as follows:

Table 16. Account object cassette properties

Field name	Syntax	Description
batchCloseTime	Integer value representing the number of minutes past midnight.	The time of day at which the currently open batches should be closed. This value is present only if it has been specified on the account configuration.

Brand

This is OfflineCard's only cassette-defined primary administrative object. One such object exists for each brand that is supported for the account.

Table 17. Brand object cassette properties

Field name	Syntax	Description
brandID	Character string, 1 to 40 bytes long	The payment card brand identifier.

OfflineCard return codes

Most error conditions raised by the OfflineCard cassette are reported exclusively through primary and secondary return codes.

- **Primary Return Codes:** Only WebSphere Commerce Payments Framework-defined primary return codes are used. Refer to the *WebSphere Commerce Payments Programmer’s Guide and Reference* for this listing.
- **Secondary Return Codes:** The majority of the secondary return codes generated by the OfflineCard cassette are defined by the WebSphere Commerce Payments Framework. Refer to the *WebSphere Commerce Payments Programmer’s Guide and Reference* for this listing. The following table lists OfflineCard-specific errors and their definitions.

RC_BUNDLE_ID_MISMATCH	20001	The OfflineCard cassette resource bundle ID mismatch.
RC_ACCOUNT_SELECT_SQL_FAILURE	21000	An SQL exception occurred while querying the OfflineCardAccount table.
RC_ACCOUNT_SELECT_CLOSE_FAILURE	21001	An SQL exception occurred while closing the query session on the OfflineCardAccount table.
RC_ACCOUNT_CREATE_ROW_FAILURE	21002	An SQL exception occurred while adding a row to the OfflineCardAccount table.
RC_ACCOUNT_CREATE_SQL_FAILURE	21003	An SQL exception occurred while inserting a row to the OfflineCardAccount table.
RC_ACCOUNT_UPDATE_ROW_FAILURE	21004	An SQL exception occurred while updating a row in the OfflineCardAccount table.
RC_ACCOUNT_UPDATE_SQL_FAILURE	21005	An SQL exception occurred while updating the OfflineCardAccount table.
RC_ACCOUNT_DELETE_ROW_FAILURE	21006	An SQL exception occurred while deleting a row from the OfflineCardAccount table.
RC_ACCOUNT_DELETE_SQL_FAILURE	21007	An SQL exception occurred while deleting the OfflineCard account table.
RC_ACCOUNT_NULL_BATCH_NUMBER	21009	An attempt to retrieve a batch for a transaction with an empty batch number was made.
RC_ACCOUNT_BRAND_ADMIN	21010	The error refers to the OfflineCardBrand object.
RC_BATCH_SELECT_SQL_FAILURE	22002	An SQL exception occurred while querying the OfflineCardBatch table.
RC_BATCH_SELECT_CLOSE_FAILURE	22003	An SQL exception occurred while closing a query session of OfflineCardBatch table.
RC_BATCH_CREATE_SQL_FAILURE	22005	An SQL exception occurred while inserting row into the OfflineCardBatch table.
RC_BATCH_UPDATE_ROW_FAILURE	22006	An SQL exception occurred while updating a row in the OfflineCardBatch table.
RC_BATCH_UPDATE_SQL_FAILURE	22007	An SQL exception occurred while updating the OfflineCardBatch table.
RC_BATCH_DELETE_ROW_FAILURE	22008	An SQL exception occurred while deleting a row from the OfflineCardBatch table.
RC_BATCH_DELETE_SQL_FAILURE	22009	An SQL exception occurred while deleting rows from the OfflineCardBatch table.
RC_BATCH_NULL_ORDER_FOR_PAYMENT	22010	A payment’s order cannot be NULL.
RC_BATCH_NULL_PAYMENT	22011	A payment object cannot be empty within a batch.
RC_BATCH_BAD_BATCH_IN_PAYMENT	22012	The batch number referenced in the payment object is different from that in the associated batch object. .

RC_BATCH_NULL_ORDER_FOR_CREDIT	22013	A credit's order cannot be NULL.
RC_BATCH_NULL_CREDIT	22014	A credit object cannot be empty within a batch.
RC_BATCH_BAD_BATCH_IN_CREDIT	22015	The batch number referenced in the credit object is different fromthat in the associated batch object.
RC_BATCH_PURGE_INCOMPLETE	22016	The batch purge operation is incomplete.
RC_ORDER_SELECT_ORDER_MISSING	23001	An SQL query for an order returns nothing.
RC_ORDER_SELECT_SQL_FAILURE	23002	An SQL query for orders failed.
RC_ORDER_SELECT_CLOSE_FAILURE	23003	An SQL exception occurred while closing an order query session.
RC_ORDER_CREATE_ROW_FAILURE	23004	An SQL exception occurred while inserting a row into the OfflineCardOrder table.
RC_ORDER_CREATE_SQL_FAILURE	23005	An SQL exception occurred while inserting rows into the OfflineCardOrder table.
RC_ORDER_UPDATE_ROW_FAILURE	23006	An SQL exception occurred while updating a row in the OfflineCardOrder table.
RC_ORDER_UPDATE_SQL_FAILURE	23007	An SQL exception occurred while updating rows in the OfflineCardOrder table.
RC_ORDER_DELETE_ROW_FAILURE	23008	An SQL exception occurred while deleting a row from the OfflineCardOrder table.
RC_ORDER_DELETE_SQL_FAILURE	23009	An SQL exception occurred while deleting rows from the OfflineCardOrder table.
RC_PAYMENT_SELECT_ROW_FAILURE	24000	An SQL exception occurred while querying a row from the OfflineCardPayment table.
RC_PAYMENT_SELECT_PAYMENT_MISSING	24001	An error occurred while querying the OfflineCardPayment table.
RC_PAYMENT_SELECT_SQL_FAILURE	24002	An SQL exception occurred while querying the OfflineCardPayment table.
RC_PAYMENT_SELECT_CLOSE_FAILURE	24003	An SQL exception occurred while closing a query session.
RC_PAYMENT_CREATE_ROW_FAILURE	24004	An SQL exception occurred while inserting a row into the OfflineCardPayment table.
RC_PAYMENT_CREATE_SQL_FAILURE	24005	An SQL exception occurred while inserting rows into the OfflineCardPayment table.
RC_PAYMENT_UPDATE_SQL_FAILURE	24007	An SQL exception occurred while updating rows in the OfflineCardPayment table.
RC_PAYMENT_DELETE_SQL_FAILURE	24009	An SQL exception occurred while deleting rows from the OfflineCardPayment table.
RC_CREDIT_SELECT_CREDIT_MISSING	25001	An error occurred while querying the OfflineCardCredit table.
RC_CREDIT_SELECT_SQL_FAILURE	25002	An SQL exception occurred while querying the OfflineCardCredit table.
RC_CREDIT_SELECT_CLOSE_FAILURE	25003	An SQL exception occurred while closing a query session from the OfflineCardCredit table.
RC_CREDIT_CREATE_SQL_FAILURE	25005	An SQL exception occurred while inserting rows into the OfflineCardCredit table.
RC_CREDIT_UPDATE_ROW_FAILURE	25006	An SQL exception occurred while updating a row in the OfflineCardCredit table.

RC_CREDIT_UPDATE_SQL_FAILURE	25007	An SQL exception occurred while updating rows in the OfflineCardCredit table.
RC_CREDIT_DELETE_ROW_FAILURE	25008	An SQL exception occurred while deleting a row from the OfflineCardCredit table.
RC_CREDIT_DELETE_SQL_FAILURE	25009	An SQL exception occurred while deleting rows from the OfflineCardCredit table.
RC_QUERY_ORD_SELECT_SQL_FAILURE	26000	An SQL exception occurred while querying the OFFLCORDERVIEW view.
RC_QUERY_PAY_SELECT_SQL_FAILURE	26001	An SQL exception occurred while querying the OFFLCPAYMENTVIEW view.
RC_QUERY_CRE_SELECT_SQL_FAILURE	26002	An SQL exception occurred while querying the OFFLCCREDITVIEW view.
RC_QUERY_BAT_SELECT_SQL_FAILURE	26003	An SQL exception occurred while querying the OFFLCBATCHVIEW view.
RC_QUERY_ACC_SELECT_SQL_FAILURE	26004	An SQL exception occurred while querying the OFFLCACCOUNTVIEW view.

OfflineCard error messages

All OfflineCard-specific messages are listed here. The following information is listed for each message:

Severity

Information
No user action is required.

Warning
You should check to determine whether you need to change anything.

Error The specific process or application needs to be fixed and restarted.

Severe
The WebSphere Commerce Payments must be restarted after resolving the problem.

Explanation
The condition that was detected.

User Response
A suggested action or list of steps to remedy the problem.

CEPOfflineCard1000 The OfflineCard Cassette has started.

Severity: Information

Explanation: The Cassette is now accepting requests.

User Response: None

CEPOfflineCard1001 The OfflineCard Cassette has stopped.

Severity: Information

Explanation: The Cassette is no longer accepting requests.

User Response: None.

CEPOfflineCard1002 This method has not been implemented yet. The method is *method_name*

Severity: Error

Explanation: Check with development as to the availability of this method if you have a need for it.

User Response: None.

CEPOfflineCard1003 The cassette's resource bundle ID does not match the ID passed by the framework. Expected ID is *expected_id* and framework ID is *framework_id*

Severity: Error

Explanation: . None.

User Response: Check with your developer for possible reasons as to why this happened.

CEPOfflineCard2000 An SQL exception occurred while selecting existing accounts from the OFFLINECARDACCOUNT table.

Severity: Error

Explanation: An SQL exception occurred while accessing the OFFLINECARDACCOUNT table. This could be due to an error connecting to or accessing the database, or due to an error in the content of the data.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPOfflineCard2001 An SQL exception occurred while closing a query on the OFFLINECARDACCOUNT table.

Severity: Error

Explanation: An SQL exception occurred while closing OFFLINECARDACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard2003 An SQL exception occurred while inserting a row into the OFFLINECARDACCOUNT table. The merchant number is *merchant_number*, the Account number is *account_number*, and the number of rows affected is *rows*

Severity: Information

Explanation: This insertion is only supposed to insert one row, but more than one row was inserted. This condition should not happen but if it does, contact support.

User Response: None.

CEPOfflineCard2004 An SQL exception occurred while inserting an account into the OFFLINECARDACCOUNT table. The merchant number is *merchant_number*, and account number is *account_number*.

Severity: Information

Explanation: An SQL exception occurred while inserting a row into the OFFLINECARDACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server. Also make sure that the database user is authorized to access the WebSphere Commerce Payments database.

CEPOfflineCard2005 An SQL update should have changed only one row but multiple rows were affected. The merchant number is *merchant_number*, the account number is *account_number*, and number of rows affected are *number*.

Severity: Information

Explanation: The update should have changed only one row but more than one row were affected.

User Response: None.

CEPOfflineCard2006 An SQL exception occurred while updating an existing account in the OFFLINECARDACCOUNT table. The merchant number is *merchant_number*, and account number is *account_number*.

Severity: Error

Explanation: SQL exception occurred while reading a WebSphere Commerce Payments database table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard2007 An SQL exception occurred while deleting a row from the OFFLINECARDACCOUNT table. The merchant number is *merchant_number*, the Account number is *account_number*, and the number of rows affected is *rows*

Severity: Error

Explanation: The deletion is only supposed to delete one row, but more than one row was deleted. This condition should not happen but if it does, contact support.

User Response: None

CEPOfflineCard2008 An SQL exception occurred while deleting an account from the OFFLINECARDACCOUNT table. The associated merchant number is *merchant_number* and account number is *account_number*.

Severity: Error

Explanation: An SQL exception occurred while inserting a row into the OFFLINECARDACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server. Also make sure that the database user is authorized to access the WebSphere Commerce Payments database.

CEPOfflineCard2009 Unable to close connection on port *port number* to the OfflineCard CashRegister at *hostname*

Severity: Error

Explanation: The socket connection that performs the communication between the cassette and the OfflineCard CashRegister failed to close successfully.

User Response: If the problem persists, contact your IBM service representative.

CEPOfflineCard2010 An attempt to retrieve a batch failed because the transaction did not contain a batch number. The merchant number is *merchant_number* , the account number is *account_number*, the order number is *order_number*, and transaction number is *transaction_number*.

Severity: Error

Explanation: The transaction used in the request is not a valid one. This is not expected to happen on a production system.

User Response: None.

CEPOfflineCard3000 An SQL query of OFFLINECARDBATCH table returned incorrect result.

Severity: Error

Explanation: An SQL query of OFFLINECARDBATCH table should have returned only one row but multiple rows were returned. The merchant number is *merchant_number* and batch number is *number*.

User Response: None.

CEPOfflineCard3001 An SQL query of OFFLINECARDBATCH table returned incorrect result.

Severity: Error

Explanation: An SQL query of OFFLINECARDBATCH table should have returned only one row but multiple rows were returned. The merchant number is *merchant_number* and batch number is *number*.

User Response: None.

CEPOfflineCard3002 An SQL exception occurred while querying a batch from the OFFLINECARDBATCH table. The merchant number is *merchant_number* and batch number is *number*.

Severity: Error

Explanation: An SQL exception occurred while reading the OFFLINECARDBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard3003 An SQL exception occurred while closing a query operation from the OFFLINECARDBATCH table.

Severity: Error

Explanation: An SQL exception occurred while closing the OFFLINECARDBATCH table query. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard3004 An SQL exception occurred while inserting a row into the OFFLINECARDBATCH table. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **and the number of rows affected is** *rows*

Severity: Error

Explanation: This insertion is only supposed to insert one row, but more than one row was inserted. This condition should not happen but if it does, contact support.

User Response: None.

CEPOfflineCard3005 An SQL exception occurred while updating an existing account in the OFFLINECARDBATCH table. The merchant number is *merchant_number* **and batch number is** *batch_number*.

Severity: Error

Explanation: SQL exception occurred while inserting into the OFFLINECARDBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard3006 An SQL exception occurred while updating a row from the OFFLINECARDBATCH table. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **and the number of rows affected is** *rows*

Severity: Error

Explanation: The update is only supposed to affect one row, but more than one row was updated. This condition should not happen but if it does, contact support.

User Response: None.

CEPOfflineCard3007 An SQL exception occurred while updating an existing batch in the OFFLINECARDBATCH table. The merchant number is *merchant_number* **and the batch number is** *batch_number*

Severity: Error

Explanation: An SQL exception occurred while reading the OFFLINECARDBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard3008 An SQL exception occurred while deleting a batch from the OFFLINECARDBATCH table. The merchant number is *merchant_number* **batch number is** *number* **and number of rows affected is** *rows*.

Severity: Error

Explanation: An SQL delete of OFFLINECARDBATCH table should have returned only one row but multiple rows were returned.

User Response: None.

CEPOfflineCard3009 An SQL exception occurred while deleting a batch from the OFFLINECARDBATCH table. The merchant number is *merchant_number* and batch number is *number*.

Severity: Error

Explanation: An SQL exception occurred while reading the OFFLINECARDBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard3010 The Supervisor could not retrieve an order referenced by the framework's batch payment list. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and payment number is *payment_number*

Severity: Error

Explanation: This is really a sanity check in the processing of request. A user may not see this.

User Response: Call IBM technical support.

CEPOfflineCard3011 Supervisor could not retrieve a payment referenced by the framework's batch payment list. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3012 The batch number referenced by a payment does not match the batch number of the batch that contains the payment. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the payment number is *merchant_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3013 The Supervisor could not retrieve an order referenced by the framework's batch credit list. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3014 The Supervisor could not retrieve a credit referenced by the framework's batch credit list. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3015 The batch number referenced by a credit does not match the batch number of the batch that contains the credit. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and the credit number is *credit_number*

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3016 While performing a batch purge, the DepositReversal failed. Even though this payment has not been successfully purged, purging will continue. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **the order number is** *order_number*, **the payment number is** *payment_number*, **the primary return code is** *prc*, **and the secondary return code is** *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3017 While performing a batch purge, the Supervisor failed to retrieve this payment. Even though this payment has not been successfully purged, purging will continue. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **the order number is** *order_number*, **the payment number is** *payment_number*, **the primary return code is** *prc*, **and the secondary return code is** *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3018 While performing a batch purge, a RefundReversal failed. Even though this credit has not been successfully purged, purging will continue. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **the order number is** *order_number*, **the credit number is** *credit_number*, **the primary return code is** *prc*, **and the secondary return code is** *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard3019 While performing a batch purge, the Supervisor failed to retrieve this credit. Even though this credit has not been successfully purged, purging will continue. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **the order number is** *order_number*, **the credit number is** *credit_number*, **the primary return code is** *prc*, **and the secondary return code is** *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPOfflineCard4000 An assertion failure occurred while querying a row from the OFFLINECARDORDER table. The merchant number is *merchant_number* **and the order number is** *order_number*.

Severity: Error

Explanation: An SQL query of the OFFLINECARDORDER table should have returned only one row but multiple rows were returned.

User Response: None.

CEPOfflineCard4001 An assertion failure occurred while querying a row from the OFFLINECARDORDER table. The merchant number is *merchant_number* **and the order number is** *order_number*.

Severity: Error

Explanation: An SQL query of the OFFLINECARDORDER table should have returned one row but nothing was returned.

User Response: None.

CEPOfflineCard4002 An SQL exception occurred while querying an existing order from the OFFLINECARDORDER table. The merchant number is *merchant_number* **and the order number is** *order_number*.

Severity: Error

Explanation: An SQL exception occurred while querying the OFFLINECARDORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard4003 An SQL exception occurred while closing a query order from the OFFLINECARDORDER table.

Severity: Error

Explanation: An SQL exception occurred while closing the OFFLINECARDORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard4004 An assertion failure occurred while inserting a row into the OFFLINECARDORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL insertion into the OFFLINECARDORDER table should have inserted one row but multiple rows were affected.

User Response: None.

CEPOfflineCard4005 An SQL exception occurred while inserting a new order into the OFFLINECARDORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while inserting into the OFFLINECARDORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard4006 An assertion failure occurred while updating the OFFLINECARDORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL update of the OFFLINECARDORDER table should have updated one row but affected multiple rows.

User Response: None.

CEPOfflineCard4007 An SQL exception occurred while updating an existing order in the OFFLINECARDORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while updating the OFFLINECARDORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard4008 An assertion failure occurred while removing a row from the OFFLINECARDORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL delete of the OFFLINECARDORDER table should have deleted one row but affected multiple rows.

User Response: None.

CEPOfflineCard4009 An SQL exception occurred while deleting an order from the OFFLINECARDORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while deleting an order from the OFFLINECARDORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard4010 An assertion failure occurred while closing the OFFLINECARDORDER table. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the payment number is *payment_number*, and the actual order state is *order_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a payment whose order is in the wrong state.

User Response: None.

CEPOfflineCard4011 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the credit number is *credit_number*, and the actual order state is *order_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a credit whose order is in the wrong state.

User Response: None.

CEPOfflineCard5000 An assertion failure occurred while querying the OFFLINECARDPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: While querying for a payment, it returned multiple entries. This condition should not have occurred.

User Response: Report the problem to your IBM technical support.

CEPOfflineCard5001 An assertion failure occurred while querying the OFFLINECARDPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: While querying for a payment that should have returned one row, nothing was returned. This condition should not have occurred.

User Response: Report the problem to your IBM technical support.

CEPOfflineCard5002 An SQL exception occurred while querying from the OFFLINECARDPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: An SQL exception occurred while querying from the OFFLINECARDPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard5003 An SQL exception occurred while closing a query operation from the OFFLINECARDPAYMENT table.

Severity: Error

Explanation: An SQL exception occurred while closing from the OFFLINECARDPAYMENT table query. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard5004 An assertion failure occurred while inserting a row into the OFFLINECARDPAYMENT table. The merchant number is *merchant_number***, the order number is** *order_number***, the payment number is** *payment_number***, and the number of rows affected is** *rows***.**

Severity: Error

Explanation: An SQL insertion into the OFFLINECARDPAYMENT table should have inserted one row but affected multiple rows.

User Response: None.

CEPOfflineCard5005 An SQL exception occurred while inserting a new payment into the OFFLINECARDPAYMENT table.

Severity: Error

Explanation: An SQL exception occurred while inserting a new payment into from the OFFLINECARDPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard5006 An assertion failure occurred while updating a row in the OFFLINECARDPAYMENT table. The merchant number is *merchant_number***, the order number is** *order_number***, the payment number is** *payment_number***, and the number of rows affected is** *rows***.**

Severity: Error

Explanation: While updating a payment that should have affected one row, multiple rows were returned. This condition should not have occurred.

User Response: None.

CEPOfflineCard5007 An SQL exception occurred while updating an existing payment in the OFFLINECARDPAYMENT table. The merchant number is *merchant_number***, the order number is** *order_number***, and the payment number is** *payment_number***.**

Severity: Error

Explanation: An SQL exception occurred while updating the OFFLINECARDPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard5008 An assertion failure occurred while deleting a row from the OFFLINECARDPAYMENT table. The merchant number is *merchant_number***, the order number is** *order_number***, the payment number is** *payment_number***, and the number of rows affected is** *rows***.**

Severity: Error

Explanation: While deleting a payment that should have affected one row, multiple rows were returned. This condition should not have occurred.

User Response: None.

CEPOfflineCard5009 An SQL exception occurred while deleting a payment from the OFFLINECARDPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard5010 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the payment number is *payment_number*, and the actual credit state is *credit_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a payment in the wrong state.

User Response: None.

CEPOfflineCard6000 An assertion failure occurred while querying the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: While querying the OFFLINECARDCREDIT table, it was supposed to return one row but it returned multiple entries.

User Response: None.

CEPOfflineCard6001 An assertion failure occurred while querying the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: While querying the OFFLINECARDCREDIT table, it should have returned one row but nothing was returned.

User Response: None.

CEPOfflineCard6002 An SQL exception occurred while querying a credit from the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6003 An SQL exception occurred while closing a query on the OFFLINECARDCREDIT table.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6004 An assertion failure occurred while querying the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, the credit number is *credit_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: While querying the OFFLINECARDCREDIT table, it should have returned one row but multiple rows are returned.

User Response: None.

CEPOfflineCard6005 An SQL exception occurred while inserting a new credit into the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6006 An assertion failure occurred while updating the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, the credit number is *credit_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: While updating the OFFLINECARDCREDIT table, it should have updated one row but multiple rows are affected.

User Response: None.

CEPOfflineCard6007 An SQL exception occurred while updating an existing credit in the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6008 An assertion failure occurred while removing a row from the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, the credit number is *credit_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: While deleting a row from the OFFLINECARDCREDIT table, it should have deleted one row but multiple rows were affected.

User Response: None.

CEPOfflineCard6009 An SQL exception occurred while deleting a credit from the OFFLINECARDCREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6010 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, the credit number is *credit_number*, and the actual credit state is *credit_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a credit in the wrong state.

User Response: None.

CEPOfflineCard6100 An SQL exception occurred while inserting a brand into the OFFLINECARDBRAND table. The merchant number is *merchant_number*, the account number is *account_number*, and the brand is *brand_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6101 An SQL exception occurred while updating a brand in the OFFLINECARDBRAND table. The merchant number is *merchant_number*, the account number is *account_number*, and the brand is *brand_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6102 An SQL exception occurred while deleting a brand from the OFFLINECARDBRAND table. The merchant number is *merchant_number*, the account number is *account_number*, and the brand is *brand_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard6103 An SQL exception occurred while querying a brand from the OFFLINECARDBRAND table.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard7000 An SQL exception occurred while processing the output from query on cassette orders. The table queried is OFFLINECARDORDER.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard7001 An SQL exception occurred while processing the output from query on cassette payments. The table queried is OFFLINECARDPAYMENT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard7002 An SQL exception occurred while processing the output from query on cassette credits. The table queried is OFFLINECARDCREDIT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard7003 An SQL exception occurred while processing the output from query on cassette batches. The table queried is OFFLINECARDBATCH.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCard7004 An SQL exception occurred while processing the output from query on cassette accounts. The table queried is OFFLINECARDACCOUNT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPOfflineCardPRC3SRC1015B The card number was not specified.

Severity:

Explanation: Return code pair - PRC_PARAMETER_NOT_FOUND, RC_CASSETTE_PAN.

User Response:

CEPOfflineCardPRC4SRC10008B The avs code is too short.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_SHORT, RC_CASSETTE_AVSCODE.

User Response:

CEPOfflineCardPRC5SRC10008B The avs code is too long.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_LONG, RC_CASSETTE_AVSCODE.

User Response:

CEPOfflineCardPRC6SRC10008B The avs code does not have the required format.

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_AVSCODE.

User Response:

Appendix B. CustomOffline cassette supplement

Like the OfflineCard cassette, the CustomOffline cassette is an instance of a *passive* cassette. As previously noted in Appendix A, “OfflineCard cassette supplement” on page 45, passive cassettes record events that have already happened outside of the WebSphere Commerce Payments within the WebSphere Commerce Payments object model. Transactions are recorded and maintained only in the WebSphere Commerce Payments database -- there is no back-end financial system with which a passive cassette communicates. Using the above example, an Approve command issued through the CustomOffline cassette records the results of an approval decision which the merchant made manually, possibly with the help of some third party to evaluate the overall risk associated with the purchase request.

This appendix describes the WebSphere Commerce Payments CustomOffline cassette, which allows merchants to use the WebSphere Commerce Payments for managing information surrounding *manual* payment transactions, such as Collect On Delivery (COD), ″Bill Me Later,″ or any other merchant-defined method for manually collecting payments. The tutorial describes how to configure both COD and BillMe. We will suggest a method for mapping typical manual operations to WebSphere Commerce Payments actions. However, regardless of the manual payment method, the merchant will ultimately decide how to map real-world actions into the WebSphere Commerce Payments. For example, the merchant will decide when to:

- Mark payments as being approved or deposited
- Create a new credit
- Close a batch

Using the CustomOffline cassette, the merchant can:

- Collect some general information about the buyer in the WebSphere Commerce Payments through their online store using a standard *buy* page. This information is entered into the WebSphere Commerce Payments using the AcceptPayment command, which stores the data in a new Order object. Along with the cassette-defined payment information (such as, billing address), the cassette allows the merchant to store up to two pieces of his own data with the Order.
- Retrieve that information through the WebSphere Commerce Payments user interface for later use in approving and collecting payments.
- Record the state of the manual transactions in the WebSphere Commerce Payments. For example, once the merchant has determined that the risk associated with a given purchase is acceptable, the merchant would use the Approve button on the user interface Order window to record the fact that he approves the purchase. Transaction recording can be performed for all payment-, credit- and batch-oriented operations.
- Manage daily batches using the WebSphere Commerce Payments’s user interface and reports.
- Perform all of these operations using the same interfaces as they use for online transactions.

Getting started

Use the information here to configure the CustomOffline cassette. For information about customizing your user interface for CustomOffline cassette see “Customizing the WebSphere Commerce Payments user interface” on page 43.

For workstation platforms, you should have completed the following at this point:

- Installed and configured the WebSphere Commerce Payments Framework
- Started the Web server
- Started WebSphere Application Server
- Started WebSphere Commerce Payments Application Server using the WebSphere administrative console
- Initialized WebSphere Commerce Payments from the IBMPayServer command file in the install directory

  Note: WebSphere Application Server, Version 3.0.2 does not start automatically. It must be started separately.
- Defined a WebSphere Commerce Payments user with administrative authority
- Created a merchant and Merchant Administrator for that merchant

If you are using an iSeries system, at this point you should have completed the following:

- Installed the WebSphere Commerce Payments framework
- Created a WebSphere Commerce Payments instance
- Added a cassette to the WebSphere Commerce Payments instance on the iSeries system
- Started the WebSphere Commerce Payments instance (which automatically starts the Web server, WebSphere Application Server and the Payment Servlet)
- Defined a WebSphere Commerce Payments user with administrative authority
- Created a merchant and Merchant administrator for that merchant

To configure a cassette, you must logon to WebSphere Commerce Payments as a Merchant administrator. For more information, see the *WebSphere Commerce Payments Installation Guide*.

Installing and configuring the CustomOffline cassette

The WebSphere Commerce Payments must be installed before the CustomOffline cassette can be installed. Installing WebSphere Commerce Payments first ensures that all prerequisite products are available for the cassette. The minimum framework that this cassette supports is WebSphere Commerce Payments Version 3.1. For information on configuring the CustomOffline cassette, see “Creating a WebSphere Commerce Payments merchant and authorizing a cassette”.

Creating a WebSphere Commerce Payments merchant and authorizing a cassette

If not already done, log into the WebSphere Commerce Payments as the Payments administrator. You now have global views and global authority. The first step in configuring the WebSphere Commerce Payments is to create a merchant and authorize that merchant to use a payment cassette:

1. From the navigation frame, click **Merchant Settings** under the Administration section.

2. From the Merchant Settings window, click **Add a Merchant**.
3. At the next window, you are prompted to enter the following (note that the italicized text *must* be entered in these fields for the tutorial):

Merchant name	Enter *Custom Store*. This is the name that you assign to the merchant. Its only function is to provide display information in the user interface.
Merchant number	Enter *987654321*. This is a number that you assign which uniquely identifies the merchant in all transaction data.
Authorized cassettes	Check the box next to the *CustomOffline cassette*. Checking this box authorizes the merchant to use this payment cassette.

4. When you have entered the requested information, click **Create Merchant** to save the merchant configuration.

If you have already created a merchant that you want to authorize this cassette to use, perform these steps:
1. Click **Merchant Settings**.
2. Click **Merchant Name**.
3. Under ″Authorized cassettes″, choose **CustomOffline**.
4. Click the **Update** button.

The cassette is now authorized to use this merchant number.

Assigning user roles

Having created:
- A user, *Pat* (this user was created in “Defining WebSphere Commerce Payments users on iSeries systems” on page 12)
- A merchant, *Custom Store*

you are ready to assign Pat’s role in the WebSphere Commerce Payments configuration.

Users must be assigned to one of the following WebSphere Commerce Payments roles:
- Payments administrator
- Merchant administrator
- Supervisor
- Clerk

Note: Users can also be assigned *no WebSphere Commerce Payments access*. Users who are granted *no WebSphere Commerce Payments access* are not able to access the WebSphere Commerce Payments. For more information on WebSphere Commerce Payments role permissions, see the Role Permissions Table in the *WebSphere Commerce Payments Programmer’s Guide and Reference*.

For the purposes of this tutorial, you will assign Pat the role of Merchant administrator for the Custom Store.
1. From the navigation frame, click **Users** under the Administration section.
2. From the Users Search window, enter the user name **Pat** and click Search.

Note: On the User Search window, you can search the WebSphere Commerce Payments database for a specific user.

3. From the Users window, click the user name **Pat**.
4. Select the merchant **Custom Store** from the available merchants in the drop-down list.
5. Select **Merchant administrator** as the user role.
6. When you have entered the requested information, click **Update** to save the user configuration.

It is important to note that in hosted environments (that is, where a Commerce Service Provider (CSP) establishes a WebSphere Commerce Payments that remotely services multiple merchants), the CSP serves as the Payments administrator, creating merchants, authorizing the merchant to use payment cassettes, and creating merchant administrators per each merchant's contract with the provider. In the hosted world, the merchant would configure his own merchant settings with Merchant administrator authority granted him by the CSP (that is, Payments administrator). This is the model we are adopting for this tutorial.

At this point, you should logoff the WebSphere Commerce Payments user interface and logon again, this time as the Merchant administrator, Pat.

Note: A merchant can use the WebSphere Commerce Payments as a non-hosted (that is, single enterprise) solution. And in a non-hosted environment, the merchant can function as *both* the Payments administrator and the Merchant administrator; in which case, the merchant could continue configuring his environment while logged in as the Payments administrator. If you will not be using the WebSphere Commerce Payments as a hosted solution, you can remain logged in as the Payments administrator (admin).

Fuzzy searches can only be performed on the user name.

Note: If you are a Merchant administrator or a Multi-Merchant administrator, you can search for users of your particular merchant numbers only.

Logging in as the Merchant administrator

To logoff and logon again:

1. Click **Logoff admin** on the navigation frame of the WebSphere Commerce Payments user interface to return to the main WebSphere Commerce Payments Logon window.
2. From the main WebSphere Commerce Payments Logon window, type the user ID `Pat` along with the password defined for Pat and click **OK**.

You are now logged in to the WebSphere Commerce Payments user interface as user Pat, with Merchant administrator authority for the Custom Store merchant. For the remainder of the tutorial, you will act as the Merchant administrator. You will notice that your view of the WebSphere Commerce Payments user interface is now limited to *merchant* administration functions; whereas, as the Payments administrator, you had a global view of both *merchant* and *WebSphere Commerce Payments* administration.

Creating an account

So far, you have defined one merchant, the Custom Store, and enabled one payment cassette, the CustomOffline cassette. Now, your first task as the Merchant administrator is to establish an *account* for the CustomOffline cassette.

An account is a relationship between the merchant and the financial institution which processes transactions for that merchant. There can be multiple accounts for each payment cassette. But for the purposes of this tutorial, you will create one account for the CustomOffline cassette.

To create an account:

1. Click **Merchant Settings** on the navigation frame of the WebSphere Commerce Payments user interface.
2. From the Merchant Settings window, click the CustomOffline cassette icon in the Custom row.
3. From the CustomOffline cassette window, click **Accounts**.
4. Click **Add an Account** on the Accounts window.
5. At the next window, you will be prompted to enter the following information:

Account name	Enter *Custom Account*. This is the name that you assign to the account. Its only function is to provide display information in the user interface.
Account number	Enter *111111111*. This is a number that you (that is, either the hosting service provider or the merchant administrator) assign which uniquely identifies the account in all transaction data.
Financial Institution name	Enter *Custom Bank*. This is the name of the financial institution with which you hold this account. Its only function is to provide display information in the user interface.
Payment method name	COD, the CustomOffline payment method name (This field is case-sensitive. In a hosted environment, you might want to customize this field. See “Customizing the WebSphere Commerce Payments user interface” on page 43 for more information.)
Batch close time	The number of minutes past midnight that the cassette will automatically try to close batches for this account. A value of 0 (zero) represents midnight. 1439 is the maximum value allowed. A null value disables automatic batch closing.

6. Click **Create account** to create an account for the CustomOffline cassette.

Managing payment processing

As the Merchant administrator, you have global *merchant* authority, which means that you can perform:

- Merchant-specific administration functions
- Payment processing functions (all)

In a real business scenario, you may choose to delegate payment processing tasks to other merchant-defined users who possess limited payment processing authorities (such as, supervisor and clerk). In this tutorial, you, as the Merchant administrator, will perform these tasks.

Having completed all of the WebSphere Commerce Payments and Merchant administration tasks necessary to begin payment processing, you are now ready to start:

- Approving orders
- Depositing payments
- Settling batches
- Issuing credits
- Viewing daily batch totals

For the purposes of this tutorial, you will use the SampleCheckout to create three orders for use in payment processing.

Creating orders using the Sample Checkout

As previously discussed, a real business environment features a buyer who creates orders using a merchant's Internet storefront and a merchant who processes payments for those orders using the WebSphere Commerce Payments. In order for you to walk through the WebSphere Commerce Payments payment processing functions, you need to create orders that require payment processing. To simulate a merchant's Internet storefront, and thus facilitate order creation, the WebSphere Commerce Payments supplies a Sample Checkout. To access the WebSphere Commerce Payments Sample Checkout and create orders:

1. Point your browser to **http://<hostname>/webapp/PaymentManager/SampleCheckout**, where `<hostname>` is the machine where the WebSphere Commerce Payments is installed.
2. At the Sample Checkout window that appears, you will be prompted to enter the following (note that the italicized text *must* be entered in these fields for the tutorial):

Merchant number	Any number to represent a merchant number
Order number	Any number to represent an order number
Amount	Any number to represent the total numeric amount of the order
Currency	Enter *US Dollar*
Payment method	Choose *COD*
Address Information	Enter an address

3. Click **Buy**.

Repeat these steps two more times so that you have three orders for which to process payments.

Approving orders

Once you have created three orders using the Sample Checkout, return to the browser window where the WebSphere Commerce Payments user interface is displayed. Point your browser once again to the WebSphere Commerce Payments URL (that is, `http://<<hostname>>/webapp/PaymentManager/`) and logon as `Pat`. Follow these steps to approve an order:

1. From the navigation frame, click **Approve** under the Payment Processing section.
2. From the Approve window, check the box next to the order you want to approve (select only one order for this exercise) and click **Approve Selected**.

3. At the Approve Results window, you will see the status of your approve request. When processing is complete, success or failure status will appear next to each order submitted for approval. When your approval is complete, click **Return to the Approve screen**.

Two orders are still awaiting your approval. You could have approved them all at once (for their full amount) by clicking **Approve All** from the Approve window. But to better demonstrate the many facets of the Approve function, you will work with each order individually.

Approving orders from the Order window

In this section, you will approve an order from the Order window (rather than from the Approve window), but you will approve only *part* of the total order amount. You may find it useful to approve only part of an order when some of the goods associated with the order are not available for delivery at order processing (for example, merchandise that is on back-order).

1. From the Approve window, click the **Order number** for one of the remaining orders awaiting approval.
2. From the Order number window, you can view order details. Click **Approve** to approve this order.
3. From the Order Number-approve window, the following information displays:

Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	This field displays zeros since no amount of the order has yet been approved. This is a read-only field.
Deposited Amount	This field displays zeros since no amount has yet been approved or deposited. This is a read-only field.
Approval Amount	This is the total amount of the order.
Authorization Reason	The authorization code returned from the manual offline authorization request process. The payment state is changed to ″Approved″.
Decline Reason	The decline reason returned from the manual offline authorization process. The payment state is changed to ″Declined″.

Change the approval amount to `3.00`. Optionally specify an authorization reason to approve the amount (or decline reason to indicate that approval was declined). Click **Approve** to approve this order for three dollars.

When approval processing has completed, you will be returned to the Order window and notified of approval success or failure. You will notice in the order details that the approved amount has been updated to reflect the three dollars we specified in the previous step.

Using the Sale function to approve orders

Because you approved only *part* of the last order you worked with, you still have two order entries in the Approve window. In this exercise, you will use the *sale* function to approve the remaining orders.

The sale function allows you to approve an order and move it directly into Deposited state, bypassing Approved state. The sale function automatically performs an Approve and a Deposit on your order payment (thus, you can think of sale as Approve with auto-deposit). Use the sale function when you want to expedite the delivery of goods to the buyer and guarantee your capture of funds (for example, when you are selling downloadable software or electronic information). However, with the sale function, you lose the ability to set the authorization reason or decline reason on the approval. Because sale provides you with the mechanism to merge Approve and Deposit into one transaction, the sale function is also useful when you are charged on a per—transaction— basis. Perform a sale as follows:

1. From the navigation frame, click **Approve** under the Payment Processing section.
2. Click **Sale All** from the Approve window.
3. At the Approve Results window, you will see a progress bar indicating the status of your sale request. When processing is complete, success or failure status will appear next to each order submitted for sale.
4. When your sale is complete, click **Return to the Approve Screen**.

Depositing payments

Deposit allows you to deposit order payments. As demonstrated in “Approving orders from the Order window” on page 20, a single order number can have multiple payments associated with it. You may see the same order number appear multiple times in the same list, each time with different payment information. To deposit a payment:

1. From the navigation frame, click **Deposit** under the Payment Processing section.
2. Check the box next to one of the payments listed and click **Deposit Selected**.
3. When processing is complete, success or failure status will appear in the Deposit Results window next to the payment submitted for deposit.
4. When your sale is complete, click **Return to the Deposit Screen**.

Note: You can deposit only *part* of a payment, in much the same way you approved part of an order:

1. From the Deposit window, click the **Payment number** for one of the payments awaiting deposit.
2. The next window is the Payment window. From the Payment window, you can view payment details. Click **Deposit** to deposit this payment.
3. On the Order Payment window, change the deposit amount to `2.00` and click **Deposit** to deposit this payment for two dollars.

Settling batches

A batch is a collection of payments and credits that are processed as a unit by a financial institution. A batch is associated with a merchant and an account. The payments that you deposited in the previous exercise will now appear in a batch. You must *settle* this batch in order to initiate processing by the financial institution. To settle a batch:

1. From the navigation frame, click **Batch Search** under the Payment Processing section.
2. At the Batch Search window, you will be prompted to enter the following information (note that for the purposes of this tutorial, you will not be entering any parameter information in the fields to narrow your search):

Merchant	The name of the merchant whose batch you are searching for. **Note:** If there are less than 500 merchants in the WebSphere Commerce Payments database, you select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, you enter the name of a merchant.
Batch Number	The number that uniquely identifies the batch within the merchant. Assigned when the payment is deposited.
State	The state of the batch: • Open • Closed
Balance Status	The balance status of this batch: • **Balanced :** the batch has been successfully balanced (that is, all totals agree). • **Out of balance:** an unsuccessful attempt has been made to balance this batch (that is, all totals do not agree).
Payment Type	Identifies the payment type, or protocol, used to place the order (for example, CustomOffline).
Batch Open Date	Use the *after* and *before* fields below to search for batches opened during the specified range in time: • **After**: Specify a date to search for all batches opened on and after this date. • **Before**: Specify a date to search for all batches opened on and before this date.
Batch Closed Date	Use the *before* and *after* fields below to search for batches closed during the specified range in time: • **After**: Specify a date to search for all batches closed on and after this date. • **Before**: Specify a date to search for all batches closed on and before this date.

3. Click **Search** to initiate a batch search.

   Note: In addition to using the *after* and *before* fields to specify a time range for the batch search (such as, 08/01/2001 to 08/15/2001). These fields can also be used to narrow search results by *excluding* certain batches from the search. For example, you could search on all batches opened *before* 08/01/2001 and all batches opened *after* 08/15/2001, thus excluding batches opened between 08/02/2001 and 08/14/2001.
4. Click the batch number to view information about the batch.
5. From the Batch window, you can view useful batch information, including the total number and amount of both payments and credits in the batch. Click **Batch Details** to see a detailed listing of all payments and credits in this batch.
6. Click **Settle** to settle the batch.
7. When processing is complete, success or failure status will appear in the Settle Results window.

When the success message displays, the settle command is complete. The financial institution is now responsible for the transfer of funds.

Note: You also have the option to delete the settled batch by clicking **Delete** on the Settle Results window. When a batch is deleted, all ancillary information about that batch (that is, payments, credits, and cassette-specific data) is deleted, as well. If you need to retain all payment data (for example, for audit purposes), then you should not delete a batch. But if you need to prune outdated information, you can exercise the **Delete** Batch option.

Issuing a credit

Credits are issued against orders and can be given for any amount. To issue a credit, you need to find the order against which you are issuing the credit:

1. From the navigation frame, click **Order Search** under the Payment Processing section.
2. On the Order Search window, you will be prompted to enter the following information (note that for the purposes of this tutorial, you will not be entering any parameter information in the fields to narrow your search):

Merchant	The name of the merchant whose order you are searching for. **Note:** If there are less than 500 merchants in the WebSphere Commerce Payments database, select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, enter the name of a merchant.
Order Number	A number assigned by the merchant that uniquely identifies the order.
State	The state of the order: • Requested • Ordered • Refundable • Canceled • Closed
Payment Type	Identifies the payment type, or protocol, used to place the order (for example, CustomOffline).
Order Date	Use the *after* and *before* fields below to search for orders opened during the specified range in time: • **After**: Specify a date to search for all orders opened on and after this date. • **Before**: Specify a date to search for all orders opened on and before this date.
Order Amount	• **Currency**: The currency used to place this order. Select the currency type from the drop-down list. • **Greater than**: Specify a value to retrieve all orders with order amounts that are greater than or equal to the value you specify. • **Less than**: Specify a value to retrieve all orders with order amounts that are less than or equal to the value you specify.

3. Click **Search** to initiate an order search.

   Note: In addition to using the *after* and *before* fields to specify a time range for the order search (such as, 08/01/2001 to 08/15/2001). These fields can also be used to narrow search results by *excluding* certain orders from the search. For example, you could search on all orders opened *before* 08/01/2001 and all orders opened *after* 08/15/2001, thus excluding orders opened between 08/02/2001 and 08/14/2001.
4. From the next window, click an order number for an order in Refundable state to view the details of that order.

5. From the Order window, click **Credit** to create a credit against this order.
6. At the Create Credit window, the following information displays:

Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	The total amount of the order that has been approved expressed in the currency used to place the order. This is a read-only field.
Deposited Amount	The total amount of the order that has been deposited expressed in the currency used to place the order. This is a read-only field.
Credit Amount	This field must be completed by the merchant administrator with the total amount to be credited to the shopper.
Authorization Reason	The authorization code returned from the manual offline authorization request process. The credit state is changed to ″Refunded″.
Decline Reason	The decline reason returned from the manual offline authorization process. The credit state is changed to ″Declined″.

   Enter the credit amount (can be *any* amount) and click **Credit**.

When credit processing has completed, you will be returned to the Order window and notified of credit success or failure. You will notice on the Order window that the newly created credit appears under **Credits** at the bottom of the window.

Viewing batch totals

The last exercise in this tutorial is viewing daily batch totals. The WebSphere Commerce Payments Reports function allows you to view *daily totals* for batches in a closed state. To generate a daily batch totals report:

1. From the navigation frame, click **Reports** under the Payment Processing section.
2. From the Reports window, click **Daily Batch Totals**.
3. At the Batch Totals Report window, you will be prompted to enter the date for which you would like a batch totals report. WebSphere Commerce Payments computes the batch totals for the date entered and generates a report. *Leave this field blank to generate a report for the current date*.
4. You will also be prompted to enter the Merchant name. If you do not enter a merchant name, a list of all of the batches for the specified date will display. If there are more than 500 batches, the first 500 batches will display and you will be prompted to narrow your search by selecting a specific merchant.
5. Click **Search** to initiate a batch report search.

The Daily Batch Totals report computes the totals for all batches that were closed on the date specified on the **Search** window. Since you did not specify a search date, the report that was generated contains the current day’s batch totals. These totals are computed on a per-currency basis, so there is one line per currency. Note that these totals cover all payments and credits made for *all payment types* (not just those made through the CustomOffline cassette).

You have just completed a day in the life of a Payments administrator and a Merchant administrator. While individual business models may vary, this tutorial outlines the basic path to establishing a working WebSphere Commerce Payments and demonstrates fundamental payment processing implemented through the WebSphere Commerce Payments. For more information on specific fields in the WebSphere Commerce Payments user interface, see the online Help.

Command reference

For each WebSphere Commerce Payments API command, the following sections describe:

- How (or whether) each optional Framework parameter is supported by the CustomOffline cassette
- Any special notes related to the CustomOffline cassette's handling of Framework parameters
- All CustomOffline-specific protocol parameters

In each table, the *Required/Optional* column has a value of **R** for required or **O** for optional.

Note: Syntax for parameters that are common to the IBM WebSphere WebSphere Commerce Payments Cassette for CyberCash follow the same syntax rules, unless otherwise noted.

Financial commands

AcceptPayment

You can use the AcceptPayment command to create an independent credit that the merchant can use to issue a refund to the customer. For example, a customer desires a refund on an order, and the payment for that order has not been made. If the merchant decides to grant the customer a refund, the merchant can use the same order that the customer placed to issue a credit. When a new order whose pre-existent state is either ORDER_RESET or ORDER_REQUESTED is created via the ACCEPTPAYMENT API, then the Order object is put in the ORDER_REFUNDABLE state. This allows the merchant to issue a refund on the order even when no payment exists.

Table 18. Required keywords for AcceptPayment command

Keyword	Required / Optional	Value
APPROVEFLAG	O	Fully supported.
PAYMENTAMOUNT	O/R	Fully supported. Required if APPROVEFLAG is set to 1.
DEPOSITFLAG	O	Fully supported.
BATCHNUMBER	O	Not allowed (must not be specified) since all batches are opened implicitly.
$METHOD	R	Indicates the manual payment method to be used. Must match one of the methods configured into the merchant's set of Manual accounts. ASCII string from 1 to 32 bytes long. No default payment method is preconfigured.
$COUNTRYCODE	O	Country code of buyer's address. Depending upon the manual payment method, this may be a billing or a shipping address. This field allows national language characters including DBCS.

Table 18. Required keywords for AcceptPayment command (continued)

$STREETADDRESS	O	Street address of buyer's address. Depending upon the manual payment method, this may be a billing or a shipping address. This field allows national language characters including DBCS.
$CITY	O	City of buyer's address. Depending upon the manual payment method, this may be a billing or a shipping address. This field allows national language characters including DBCS.
$STATEPROVINCE	O	State or province of buyer's address. Depending upon the manual payment method, this may be a billing or a shipping address. This field allows national language characters including DBCS.
$POSTALCODE	O	Postal (zip) code of buyer's address. Depending upon the manual payment method, this may be a billing or a shipping address. This field allows national language characters including DBCS.
$AUXILIARY1	O	Binary string, 0 to 254 characters long. Merchant-supplied payment data. Only the merchant software knows what is stored in this field.
$AULILIARY2	O	Binary string, 0 to 254 characters long. Merchant-supplied payment data. Only the merchant software knows what is stored in this field.
$AUTHCODE	O	Authorization code to be saved with the approved Payment. This parameter is only used if APPROVEFLAG=1 is also specified on the AcceptPayment command. Must be an ASCII character string between 1 and 64 bytes long. If $AUTHCODE is specified, then a Payment is also created in PAYMENT_APPROVED state for the Order amount and the ReferenceNumber is set to the $AUTHCODE value. If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_ AND_DECLINEREASON. If neither this parameter nor $DECLINEREASON is specified with APPROVEFLAG=1, then an approved Payment is created for the Order amount with an empty ReferenceCode.
$DECLINEREASON	O	Decline code to be saved with the declined Payment. This parameter is only used if APPROVEFLAG=1 is also specified on the AcceptPayment command. Must be an ASCII character string between 1 and 16 bytes long. If $DECLINEREASON is specified, then a Payment is also created in PAYMENT_DECLINED state and the ReferenceNumber is set to the $DECLINEREASON value. If specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON

Approve

Table 19. Required keywords for Approve command

Keyword	Required / Optional	Value
BATCHNUMBER	O	Not allowed (must not be specified) since all batches are opened implicitly.
DEPOSITFLAG	O	Fully supported.
$AUTHCODE	O	Authorization code to be saved with the approved Payment. Must be an ASCII character string between 1 and 16 bytes long. If $AUTHCODE is specified, then the Payment is created in PAYMENT_APPROVED state and the ReferenceNumber is set to the $AUTHCODE value. If neither this parameter nor $DECLINEREASON is specified, then the Payment is created in PAYMENT_APPROVED state with an empty ReferenceCode. If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON
$DECLINEREASON	O	Decline code to be saved with the declined Payment. Must be an ASCII character string between 1 and 256 bytes long. If $DECLINEREASON is specified, then the Payment is created in PAYMENT_DECLINED state and the declineReason field is set to the $DECLINEREASON value. If specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON

BatchClose

Table 20. Required keywords for BatchClose command

Keyword	Required / Optional	Value
$FIBATCHID	O	Specifies the financial institution's identifier for the batch. This parameters allows the merchant to correlate the local batch object to an entity at the financial institution. If specified, the value is saved as the fiBatchId value in the CustomOffline cassette's extension to the Batch object. Otherwise, that field is left empty. Must be an ASCII character string between 1 and 32 bytes long.

Deposit

Table 21. Required keywords for Deposit command

Keyword	Required / Optional	Value
BATCHNUMBER	O	Not allowed (must not be specified) since all batches are opened implicitly.

Refund

Table 22. Required keywords for Refund command

Keyword	Required / Optional	Value
BATCHNUMBER	O	Not allowed (must not be specified) since all batches are opened implicitly.
$AUTHCODE	O	Authorization code to be saved with the approved Credit. Must be an ASCII character string between 1 and 16 bytes long. If $AUTHCODE is specified, then the Credit is created in CREDIT_APPROVED state and the ReferenceNumber is set to the $AUTHCODE value. If neither this parameter nor $DECLINEREASON is specified, then the Credit is created in CREDIT_APPROVED state with an empty ReferenceCode. If specified with $DECLINEREASON, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON
$DECLINEREASON	O	Decline code to be saved with the declined Credit. Must be an ASCII character string between 1 and 256 bytes long. If $DECLINEREASON is specified, then the Credit is created in CREDIT_DECLINED state and the declineReason field is set to the $DECLINEREASON value. If specified with $AUTHCODE, the command will fail with PRC_INVALID_PARAMETER_COMBINATION, RC_CASSETTE_AUTHCODE_AND_DECLINEREASON

Other supported financial commands

The following financial commands are supported without any optional or protocol data parameters:

- **ApproveReversal** modifies the approved amount of a payment.
- **BatchPurge** clears out a batch and returns the Batch object to the Open state.
- **CancelOrder** moves an order into the cancelled state.
- **CloseOrder** moves an order into the closed state.
- **DepositReversal** disassociates a payment from a batch.
- **DeleteBatch** removes the specified batch from the database tables.
- **RefundReversal** voids existing credit objects.

Unsupported financial commands

The following financial commands are not supported and always return

`PRC_COMMAND_NOT_SUPPORTED, RC_NONE`

- **BatchOpen**
- **ReceivePayment**

Administration commands

CreateAccount

Table 23. Required keywords for CreateAccount command

Keyword	Required / Optional	Value
$METHOD	R	Indicates the manual payment method. Must be an ASCII string between 1 and 32 bytes long. Each account within a merchant must have a unique Method value.
$BATCHCLOSETIME	O	This parameter specifies the time of day at which the cassette should automatically close the open batches. If not specified, all batches must be explicitly closed through the BatchClose command. This value is specified as an integer representing the number of minutes past midnight (where 0 represents midnight). Valid values are 0 <= time <= 1439

ModifyAccount

Table 24. Required keywords for ModifyAccount command

Keyword	Required / Optional	Value
$BATCHCLOSETIME	R	This parameter specifies the time of day at which the cassette should automatically close the open batches. If not specified, all batches must be explicitly closed through the BatchClose command. This value is specified as an integer representing the number of minutes past midnight (where 0 represents midnight). Valid values are 0 <= time <= 1439

Other supported administration commands

The following administration commands are supported without any optional or protocol data parameters:

- **CassetteControl**
- **CreatePaySystem**
- **DeleteAccount**
- **DeletePaySystem**
- **ModifyCassette**
- **ModifyPaySystem**

Unsupported administration commands

The following administration commands are not supported and always return

`PRC_COMMAND_NOT_SUPPORTED, RC_NONE`

- **CreateMerchantCassetteObject**
- **CreateSystemCassetteObject**
- **DeleteMerchantCassetteObject**
- **DeleteSystemCassetteObject**
- **ModifyMerchantCassetteObject**
- **ModifySystemCassetteObject**

Object reference

The CustomOffline cassette object model closely reflects the generic model of the WebSphere Commerce Payments. This section describes both the financial and the administration objects and their cassette extensions to the various Framework objects.

Financial objects

The CustomOffline financial object model mirrors that of the generic model in that an CustomOffline object is defined to augment each generic financial object. The CustomOffline objects appear as extensions to the generic objects. Those extensions are as follows:

Order

Table 25. Order object cassette properties

Field name	Syntax	Description
auxiliary1	Binary string, 0 to 254 characters long.	Merchant-supplied payment data. Only the merchant software knows what is stored in this field.
auxiliary2	Binary string, 0 to 254 characters long.	Merchant-supplied payment data. Only the merchant software knows what is stored in this field.
country	Character string, 1 – 50.	ISO 3166 country code of cardholder’s country of residence or the country name. This value is only present if a non-null $COUNTRYCODE value was specified on the ACCEPTPAYMENT command.
street	Character string, 1 to 128 bytes long	Cardholder’s street address. This value is only present if a non-null $STREETADDRESS value was specified on the ACCEPTPAYMENT command.
city	Character string, 1 to 50 bytes long	Cardholder’s city of residence. This value is only present if a non-null $CITY value was specified on the ACCEPTPAYMENT command.
state	Character string, 1 to 50 bytes long	Cardholder’s state or province of residence. This value is only present if a non-null $STATEPROVINCE value was specified on the ACCEPTPAYMENT command.
postalCode	Character string, 1 to 14 bytes long	Cardholder’s postal code. This value is only present if a non-null $POSTALCODE value was specified on the ACCEPTPAYMENT command.

Payment

Table 26. Payment object cassette properties

Field name	Syntax	Description
authReason	Character string, 1 to 254 bytes long	The authorization code entered through the $AUTHCODE parameter of the Approve command. This field is present only when an authCode value exists for this approved payment.
declineReason	Character string, 1 to 254 bytes long	The decline reason entered through the $DECLINEREASON parameter of the Approve command. This field is present only when a declineReason exists for this declined payment.

Credit

Table 27. Credit object cassette properties

Field name	Syntax	Description
authReason	Character string, 1 to 254 bytes long	The authorization code entered through the $AUTHCODE parameter of the Refund command. This field is present only when an authCode value exists for this credited credit.
declineReason	Character string, 1 to 254 bytes long.	The decline reason entered through the $DECLINEREASON parameter of the Refund command. This field is present only when a declineReason exists for this void refund.

Batch

Table 28. Batch object cassette properties

Field name	Syntax	Description
FIBatchID	Character string, 1 to 32 bytes long	The financial institution's identifier for the batch as entered through the $FIBATCHID parameter on the BatchClose command. The merchant should use this identifier when contacting the financial institution with inquiries concerning this batch.

Administration objects

The CustomOffline extends one Framework administrative object - the AccountAdmin object. Only one Account is supported for each merchant. The CustomOffline Account extensions appear as extensions to the generic objects. Those extensions are as follows:

Account

The CustomOffline Account extensions appear as extensions to the generic objects. Those extensions are as follows:

Table 29. Account object cassette properties

Field name	Syntax	Description
method	Character string, 1 to 32 bytes long.	The manual payment method. For example, ″COD,″ ″BillMe,″ or some other merchant-defined method. This value is always present.
batchCloseTime	Integer value representing the number of minutes past midnight.	The time of day at which the currently open batches should be closed. This value is present only if it has been specified on the account configuration.

CustomOffline return codes

CustomOffline return codes follow.

RC_BUNDLE_ID_MISMATCH	20001	The CustomOffline cassette resource bundle ID mismatch.
RC_ACCOUNT_SELECT_SQL_FAILURE	21000	An SQL exception occurred while querying the CustomOfflineAccount table.

RC_ACCOUNT_SELECT_CLOSE_FAILURE	21001	An SQL exception occurred while closing the query session on the CustomOfflineAccount table.
RC_ACCOUNT_CREATE_ROW_FAILURE	21002	An SQL exception occurred while adding a row to the CustomOfflineAccount table.
RC_ACCOUNT_CREATE_SQL_FAILURE	21003	An SQL exception occurred while inserting a row to the CustomOfflineAccount table.
RC_ACCOUNT_UPDATE_ROW_FAILURE	21004	An SQL exception occurred while updating a row in the CustomOfflineAccount table.
RC_ACCOUNT_UPDATE_SQL_FAILURE	21005	An SQL exception occurred while updating the CustomOfflineAccount table.
RC_ACCOUNT_DELETE_ROW_FAILURE	21006	An SQL exception occurred while deleting a row from the CustomOfflineAccount table.
RC_ACCOUNT_DELETE_SQL_FAILURE	21007	An SQL exception occurred while deleting the CustomOffline account table.
RC_ACCOUNT_NULL_BATCH_NUMBER	21009	An attempt to retrieve a batch for a transaction with an empty batch number was made.
RC_BATCH_SELECT_SQL_FAILURE	22002	An SQL exception occurred while querying the CustomOfflineBatch table.
RC_BATCH_SELECT_CLOSE_FAILURE	22003	An SQL exception occurred while closing a query session of CustomOfflineBatch table.
RC_BATCH_CREATE_SQL_FAILURE	22005	An SQL exception occurred while inserting row into the CustomOfflineBatch table.
RC_BATCH_UPDATE_ROW_FAILURE	22006	An SQL exception occurred while updating a row in the CustomOfflineBatch table.
RC_BATCH_UPDATE_SQL_FAILURE	22007	An SQL exception occurred while updating the CustomOfflineBatch table.
RC_BATCH_DELETE_ROW_FAILURE	22008	An SQL exception occurred while deleting a row from the CustomOfflineBatch table.
RC_BATCH_DELETE_SQL_FAILURE	22009	An SQL exception occurred while deleting rows from the CustomOfflineBatch table.
RC_BATCH_NULL_ORDER_FOR_PAYMENT	22010	A payment's order cannot be NULL.
RC_BATCH_NULL_PAYMENT	22011	A payment object cannot be empty within a batch.
RC_BATCH_BAD_BATCH_IN_PAYMENT	22012	The batch number referenced in the payment object is different from that in the associated batch object. .
RC_BATCH_NULL_ORDER_FOR_CREDIT	22013	A credit's order cannot be NULL.
RC_BATCH_NULL_CREDIT	22014	A credit object cannot be empty within a batch.
RC_BATCH_BAD_BATCH_IN_CREDIT	22015	The batch number referenced in the credit object is different fromthat in the associated batch object.
RC_BATCH_PURGE_INCOMPLETE	22016	The batch purge operation is incomplete.
RC_ORDER_SELECT_ORDER_MISSING	23001	An SQL query for an order returns nothing.
RC_ORDER_SELECT_SQL_FAILURE	23002	An SQL query for orders failed.
RC_ORDER_SELECT_CLOSE_FAILURE	23003	An SQL exception occurred while closing an order query session.
RC_ORDER_CREATE_ROW_FAILURE	23004	An SQL exception occurred while inserting a row into the CustomOfflineOrder table.

RC_ORDER_CREATE_SQL_FAILURE	23005	An SQL exception occurred while inserting rows into the CustomOfflineOrder table.
RC_ORDER_UPDATE_ROW_FAILURE	23006	An SQL exception occurred while updating a row in the CustomOfflineOrder table.
RC_ORDER_UPDATE_SQL_FAILURE	23007	An SQL exception occurred while updating rows in the CustomOfflineOrder table.
RC_ORDER_DELETE_ROW_FAILURE	23008	An SQL exception occurred while deleting a row from the CustomOfflineOrder table.
RC_ORDER_DELETE_SQL_FAILURE	23009	An SQL exception occurred while deleting rows from the CustomOfflineOrder table.
RC_PAYMENT_SELECT_ROW_FAILURE	24000	An SQL exception occurred while querying a row from the CustomOfflinePayment table.
RC_PAYMENT_SELECT_PAYMENT_MISSING	24001	An error occurred while querying the CustomOfflinePayment table.
RC_PAYMENT_SELECT_SQL_FAILURE	24002	An SQL exception occurred while querying the CustomOfflinePayment table.
RC_PAYMENT_SELECT_CLOSE_FAILURE	24003	An SQL exception occurred while closing a query session.
RC_PAYMENT_CREATE_ROW_FAILURE	24004	An SQL exception occurred while inserting a row into the CustomOfflinePayment table.
RC_PAYMENT_CREATE_SQL_FAILURE	24005	An SQL exception occurred while inserting rows into the CustomOfflinePayment table.
RC_PAYMENT_UPDATE_SQL_FAILURE	24007	An SQL exception occurred while updating rows in the CustomOfflinePayment table.
RC_PAYMENT_DELETE_SQL_FAILURE	24009	An SQL exception occurred while deleting rows from the CustomOfflinePayment table.
RC_CREDIT_SELECT_CREDIT_MISSING	25001	An error occurred while querying the CustomOfflineCredit table.
RC_CREDIT_SELECT_SQL_FAILURE	25002	An SQL exception occurred while querying the CustomOfflineCredit table.
RC_CREDIT_SELECT_CLOSE_FAILURE	25003	An SQL exception occurred while closing a query session from the CustomOfflineCredit table.
RC_CREDIT_CREATE_SQL_FAILURE	25005	An SQL exception occurred while inserting rows into the CustomOfflineCredit table.
RC_CREDIT_UPDATE_ROW_FAILURE	25006	An SQL exception occurred while updating a row in the CustomOfflineCredit table.
RC_CREDIT_UPDATE_SQL_FAILURE	25007	An SQL exception occurred while updating rows in the CustomOfflineCredit table.
RC_CREDIT_DELETE_ROW_FAILURE	25008	An SQL exception occurred while deleting a row from the CustomOfflineCredit table.
RC_CREDIT_DELETE_SQL_FAILURE	25009	An SQL exception occurred while deleting rows from the CustomOfflineCredit table.
RC_QUERY_ORD_SELECT_SQL_FAILURE	26000	An SQL exception occurred while querying the CUSTOMORDERVIEW view.
RC_QUERY_PAY_SELECT_SQL_FAILURE	26001	An SQL exception occurred while querying the CUSTOMPAYMENTVIEW view.
RC_QUERY_CRE_SELECT_SQL_FAILURE	26002	An SQL exception occurred while querying the CUSTOMCREDITVIEW view.

RC_QUERY_BAT_SELECT_SQL_FAILURE	26003	An SQL exception occurred while querying the CUSTOMBATCHVIEW view.
RC_QUERY_ACC_SELECT_SQL_FAILURE	26004	An SQL exception occurred while querying the CUSTOMACCOUNTVIEW view.

CustomOffline error messages

CustomOffline error messages follow.

CEPCustomOffline1000 The CustomOffline Cassette has started.

Severity: Information

Explanation: The Cassette is now accepting requests.

User Response: None

CEPCustomOffline1001 The CustomOffline Cassette has stopped.

Severity: Information

Explanation: The Cassette is no longer accepting requests.

User Response: None.

CEPCustomOffline1002 This method has not been implemented yet. The method is *method_name*

Severity: Error

Explanation: Check with development as to the availability of this method if you have a need for it.

User Response: None.

CEPCustomOffline1003 The cassette's resource bundle ID does not match the ID passed by the framework. Expected ID is *expected_id* **and framework ID is** *framework_id*

Severity: Error

Explanation: . None.

User Response: Check with your developer for possible reasons as to why this happened.

CEPCustomOffline2000 An SQL exception occurred while selecting existing accounts from the CUSTOMOFFLINEACCOUNT table.

Severity: Error

Explanation: An SQL exception occurred while accessing the CUSTOMOFFLINEACCOUNT table. This could be due to an error connecting to or accessing the database, or due to an error in the content of the data.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPCustomOffline2001 An SQL exception occurred while closing a query on the CUSTOMOFFLINEACCOUNT table.

Severity: Error

Explanation: An SQL exception occurred while closing CUSTOMOFFLINEACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline2003 An SQL exception occurred while inserting a row into the CUSTOMOFFLINEACCOUNT table. The merchant number is *merchant_number***, the Account number is** *account_number***, and the number of rows affected is** *rows*

Severity: Information

Explanation: This insertion is only supposed to insert one row, but more than one row was inserted. This condition should not happen but if it does, contact support.

User Response: None.

CEPCustomOffline2004 An SQL exception occurred while inserting an account into the CUSTOMOFFLINEACCOUNT table. The merchant number is *merchant_number*, and account number is *account_number*.

Severity: Information

Explanation: An SQL exception occurred while inserting a row into the CUSTOMOFFLINEACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server. Also make sure that the database user is authorized to access the WebSphere Commerce Payments database.

CEPCustomOffline2005 An SQL update should have changed only one row but multiple rows were affected. The merchant number is *merchant_number*, the account number is *account_number*, and number of rows affected are *number*.

Severity: Information

Explanation: The update should have changed only one row but more than one row were affected.

User Response: None.

CEPCustomOffline2006 An SQL exception occurred while updating an existing account in the CUSTOMOFFLINEACCOUNT table. The merchant number is *merchant_number*, and account number is *account_number*.

Severity: Error

Explanation: SQL exception occurred while reading a WebSphere Commerce Payments database table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline2007 An SQL exception occurred while deleting a row from the CUSTOMOFFLINEACCOUNT table. The merchant number is *merchant_number*, the Account number is *account_number*, and the number of rows affected is *rows*

Severity: Error

Explanation: The deletion is only supposed to delete one row, but more than one row was deleted. This condition should not happen but if it does, contact support.

User Response: None

CEPCustomOffline2008 An SQL exception occurred while deleting an account from the CUSTOMOFFLINEACCOUNT table. The associated merchant number is *merchant_number* and account number is *account_number*.

Severity: Error

Explanation: An SQL exception occurred while inserting a row into the CUSTOMOFFLINEACCOUNT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server. Also make sure that the database user is authorized to access the WebSphere Commerce Payments database.

CEPCustomOffline2009 Unable to close connection on port *port number* to the CustomOffline CashRegister at *hostname*

Severity: Error

Explanation: The socket connection that performs the communication between the cassette and the CustomOffline CashRegister failed to close successfully.

User Response: If the problem persists, contact your IBM service representative.

CEPCustomOffline2010 An attempt to retrieve a batch failed because the transaction did not contain a batch number. The merchant number is *merchant_number* , the account number is *account_number*, the order number is *order_number*, and transaction number is *transaction_number*.

Severity: Error

Explanation: The transaction used in the request is not a valid one. This is not expected to happen on a production system.

User Response: None.

CEPCustomOffline3000 An SQL query of CUSTOMOFFLINEBATCH table returned incorrect result.

Severity: Error

Explanation: An SQL query of CUSTOMOFFLINEBATCH table should have returned only one row but multiple rows were returned. The merchant number is *merchant_number* and batch number is *number*.

User Response: None.

CEPCustomOffline3001 An SQL query of CUSTOMOFFLINEBATCH table returned incorrect result.

Severity: Error

Explanation: An SQL query of CUSTOMOFFLINEBATCH table should have returned only one row but multiple rows were returned. The merchant number is *merchant_number* and batch number is *number*.

User Response: None.

CEPCustomOffline3002 An SQL exception occurred while querying a batch from the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number* and batch number is *number*.

Severity: Error

Explanation: An SQL exception occurred while reading the CUSTOMOFFLINEBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline3003 An SQL exception occurred while closing a query operation from the CUSTOMOFFLINEBATCH table.

Severity: Error

Explanation: An SQL exception occurred while closing the CUSTOMOFFLINEBATCH table query. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline3004 An SQL exception occurred while inserting a row into the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number*, the batch number is *batch_number*, and the number of rows affected is *rows*

Severity: Error

Explanation: This insertion is only supposed to insert one row, but more than one row was inserted. This condition should not happen but if it does, contact support.

User Response: None.

CEPCustomOffline3005 An SQL exception occurred while updating an existing account in the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number* and batch number is *batch_number*.

Severity: Error

Explanation: SQL exception occurred while inserting into the CUSTOMOFFLINEBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline3006 An SQL exception occurred while updating a row from the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number*, the batch number is *batch_number*, and the number of rows affected is *rows*

Severity: Error

Explanation: The update is only supposed to affect one row, but more than one row was updated. This condition should not happen but if it does, contact support.

User Response: None.

CEPCustomOffline3007 An SQL exception occurred while updating an existing batch in the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number* and the batch number is *batch_number*

Severity: Error

Explanation: An SQL exception occurred while reading the CUSTOMOFFLINEBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline3008 An SQL exception occurred while deleting a batch from the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number* batch number is *number* and number of rows affected is *rows*.

Severity: Error

Explanation: An SQL delete of CUSTOMOFFLINEBATCH table should have returned only one row but multiple rows were returned.

User Response: None.

CEPCustomOffline3009 An SQL exception occurred while deleting a batch from the CUSTOMOFFLINEBATCH table. The merchant number is *merchant_number* and batch number is *number*.

Severity: Error

Explanation: An SQL exception occurred while reading the CUSTOMOFFLINEBATCH table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline3010 The Supervisor could not retrieve an order referenced by the framework's batch payment list. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and payment number is *payment_number*

Severity: Error

Explanation: This is really a sanity check in the processing of request. A user may not see this.

User Response: Call IBM technical support.

CEPCustomOffline3011 Supervisor could not retrieve a payment referenced by the framework's batch payment list. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3012 The batch number referenced by a payment does not match the batch number of the batch that contains the payment. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the payment number is *merchant_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3013 The Supervisor could not retrieve an order referenced by the framework's batch credit list. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3014 The Supervisor could not retrieve a credit referenced by the framework's batch credit list. The merchant number is *merchant_number*, the batch number is *batch_number*. the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3015 The batch number referenced by a credit does not match the batch number of the batch that contains the credit. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, and the credit number is *credit_number*

Severity: Error

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3016 While performing a batch purge, the DepositReversal failed. Even though this payment has not been successfully purged, purging will continue. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, the payment number is *payment_number*, the primary return code is *prc*, and the secondary return code is *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3017 While performing a batch purge, the Supervisor failed to retrieve this payment. Even though this payment has not been successfully purged, purging will continue. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, the payment number is *payment_number*, the primary return code is *prc*, and the secondary return code is *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3018 While performing a batch purge, a RefundReversal failed. Even though this credit has not been successfully purged, purging will continue. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, the credit number is *credit_number*, the primary return code is *prc*, and the secondary return code is *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline3019 While performing a batch purge, the Supervisor failed to retrieve this credit. Even though this credit has not been successfully purged, purging will continue. The merchant number is *merchant_number*, the batch number is *batch_number*, the order number is *order_number*, the credit number is *credit_number*, the primary return code is *prc*, and the secondary return code is *src*.

Severity: Warning

Explanation: This is an internal test to assure that further processing does not fail.

User Response: None.

CEPCustomOffline4000 An assertion failure occurred while querying a row from the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL query of the CUSTOMOFFLINEORDER table should have returned only one row but multiple rows were returned.

User Response: None.

CEPCustomOffline4001 An assertion failure occurred while querying a row from the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL query of the CUSTOMOFFLINEORDER table should have returned one row but nothing was returned.

User Response: None.

CEPCustomOffline4002 An SQL exception occurred while querying an existing order from the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while querying the CUSTOMOFFLINEORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline4003 An SQL exception occurred while closing a query order from the CUSTOMOFFLINEORDER table.

Severity: Error

Explanation: An SQL exception occurred while closing the CUSTOMOFFLINEORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline4004 An assertion failure occurred while inserting a row into the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL insertion into the CUSTOMOFFLINEORDER table should have inserted one row but multiple rows were affected.

User Response: None.

CEPCustomOffline4005 An SQL exception occurred while inserting a new order into the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while inserting into the CUSTOMOFFLINEORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline4006 An assertion failure occurred while updating the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL update of the CUSTOMOFFLINEORDER table should have updated one row but affected multiple rows.

User Response: None.

CEPCustomOffline4007 An SQL exception occurred while updating an existing order in the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while updating the CUSTOMOFFLINEORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline4008 An assertion failure occurred while removing a row from the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number*, the order number is *order_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL delete of the CUSTOMOFFLINEORDER table should have deleted one row but affected multiple rows.

User Response: None.

CEPCustomOffline4009 An SQL exception occurred while deleting an order from the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number* and the order number is *order_number*.

Severity: Error

Explanation: An SQL exception occurred while deleting an order from the CUSTOMOFFLINEORDER table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline4010 An assertion failure occurred while closing the CUSTOMOFFLINEORDER table. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the payment number is *payment_number*, and the actual order state is *order_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a payment whose order is in the wrong state.

User Response: None.

CEPCustomOffline4011 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the credit number is *credit_number*, and the actual order state is *order_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a credit whose order is in the wrong state.

User Response: None.

CEPCustomOffline5000 An assertion failure occurred while querying the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: While querying for a payment, it returned multiple entries. This condition should not have occurred.

User Response: Report the problem to your IBM technical support.

CEPCustomOffline5001 An assertion failure occurred while querying the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: While querying for a payment that should have returned one row, nothing was returned. This condition should not have occurred.

User Response: Report the problem to your IBM technical support.

CEPCustomOffline5002 An SQL exception occurred while querying from the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: An SQL exception occurred while querying from the CUSTOMOFFLINEPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline5003 An SQL exception occurred while closing a query operation from the CUSTOMOFFLINEPAYMENT table.

Severity: Error

Explanation: An SQL exception occurred while closing from the CUSTOMOFFLINEPAYMENT table query. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline5004 An assertion failure occurred while inserting a row into the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, the payment number is *payment_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: An SQL insertion into the CUSTOMOFFLINEPAYMENT table should have inserted one row but affected multiple rows.

User Response: None.

CEPCustomOffline5005 An SQL exception occurred while inserting a new payment into the CUSTOMOFFLINEPAYMENT table.

Severity: Error

Explanation: An SQL exception occurred while inserting a new payment into from the CUSTOMOFFLINEPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline5006 An assertion failure occurred while updating a row in the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, the payment number is *payment_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: While updating a payment that should have affected one row, multiple rows were returned. This condition should not have occurred.

User Response: None.

CEPCustomOffline5007 An SQL exception occurred while updating an existing payment in the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: An SQL exception occurred while updating the CUSTOMOFFLINEPAYMENT table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline5008 An assertion failure occurred while deleting a row from the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, the payment number is *payment_number*, and the number of rows affected is *rows*.

Severity: Error

Explanation: While deleting a payment that should have affected one row, multiple rows were returned. This condition should not have occurred.

User Response: None.

CEPCustomOffline5009 An SQL exception occurred while deleting a payment from the CUSTOMOFFLINEPAYMENT table. The merchant number is *merchant_number*, the order number is *order_number*, and the payment number is *payment_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline5010 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, the order number is *order_number*, the batch number is *batch_number*, the payment number is *payment_number*, and the actual credit state is *credit_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a payment in the wrong state.

User Response: None.

CEPCustomOffline6000 An assertion failure occurred while querying the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number*, the order number is *order_number*, and the credit number is *credit_number*.

Severity: Error

Explanation: While querying the CUSTOMOFFLINECREDIT table, it was supposed to return one row but it returned multiple entries.

User Response: None.

CEPCustomOffline6001 An assertion failure occurred while querying the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number***, the order number is** *order_number***, and the credit number is** *credit_number***.**

Severity: Error

Explanation: While querying the CUSTOMOFFLINECREDIT table, it should have returned one row but nothing was returned.

User Response: None.

CEPCustomOffline6002 An SQL exception occurred while querying a credit from the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number***, the order number is** *order_number***, and the credit number is** *credit_number***.**

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline6003 An SQL exception occurred while closing a query on the CUSTOMOFFLINECREDIT table.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline6004 An assertion failure occurred while querying the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number***, the order number is** *order_number***, the credit number is** *credit_number***, and the number of rows affected is** *rows***.**

Severity: Error

Explanation: While querying the CUSTOMOFFLINECREDIT table, it should have returned one row but multiple rows are returned.

User Response: None.

CEPCustomOffline6005 An SQL exception occurred while inserting a new credit into the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number***, the order number is** *order_number***, and the credit number is** *credit_number***.**

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline6006 An assertion failure occurred while updating the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number***, the order number is** *order_number***, the credit number is** *credit_number***, and the number of rows affected is** *rows***.**

Severity: Error

Explanation: While updating the CUSTOMOFFLINECREDIT table, it should have updated one row but multiple rows are affected.

User Response: None.

CEPCustomOffline6007 An SQL exception occurred while updating an existing credit in the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number*, **the order number is** *order_number*, **and the credit number is** *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline6008 An assertion failure occurred while removing a row from the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number*, **the order number is** *order_number*, **the credit number is** *credit_number*, **and the number of rows affected is** *rows*.

Severity: Error

Explanation: While deleting a row from the CUSTOMOFFLINECREDIT table, it should have deleted one row but multiple rows were affected.

User Response: None.

CEPCustomOffline6009 An SQL exception occurred while deleting a credit from the CUSTOMOFFLINECREDIT table. The merchant number is *merchant_number*, **the order number is** *order_number*, **and the credit number is** *credit_number*.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline6010 An assertion failure occurred while closing a batch. The merchant number is *merchant_number*, **the batch number is** *batch_number*, **the order number is** *order_number*, **the credit number is** *credit_number*, **and the actual credit state is** *credit_state*.

Severity: Error

Explanation: While closing a batch, it was discovered that the batch contained a credit in the wrong state.

User Response: None.

CEPCustomOffline7000 An SQL exception occurred while processing the output from query on cassette orders. The table queried is CUSTOMOFFLINEORDER.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline7001 An SQL exception occurred while processing the output from query on cassette payments. The table queried is CUSTOMOFFLINEPAYMENT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline7002 An SQL exception occurred while processing the output from query on cassette credits. The table queried is CUSTOMOFFLINECREDIT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline7003 An SQL exception occurred while processing the output from query on cassette batches. The table queried is CUSTOMOFFLINEBATCH.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOffline7004 An SQL exception occurred while processing the output from query on cassette accounts. The table queried is CUSTOMOFFLINEACCOUNT.

Severity: Error

Explanation: The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPCustomOfflinePRC3SRC1100M The payment method name was not specified.

Severity:

Explanation: Return code pair - PRC_PARAMETER_NOT_FOUND, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC3SRC10000B The method was not specified.

Severity:

Explanation: Return code pair - PRC_PARAMETER_NOT_FOUND, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC3SRC10005B The expiration date was not specified.

Severity: Error

Explanation: Expiration date was not found.

User Response: Specify the expiration date.

CEPCustomOfflinePRC4SRC10000B The method is too short.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_SHORT, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC4SRC10001B The auxiliary1 value is too short.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_SHORT, RC_CASSETTE_AUXILIARY1.

User Response:

CEPCustomOfflinePRC4PRC10002B The auxiliary1 value is too short.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_SHORT, RC_CASSETTE_AUXILIARY1.

User Response:

CEPCustomOfflinePRC5SRC1100M The payment method name is too long.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_LONG, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC5SRC10000B The method is too long.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_LONG, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC5SRC10001B The auxiliary1 value is too long.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_LONG, RC_CASSETTE_AUXILIARY1.

User Response:

CEPCustomOfflinePRC5PRC10002B The auxiliary2 value is too long.

Severity:

Explanation: Return code pair - PRC_PARAMETER_TOO_LONG, RC_CASSETTE_AUXILIARY2.

User Response:

CEPCustomOfflinePRC6SRC1099M The batch close time does not have the required format.

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_BATCHCLOSETIME.

User Response:

CEPCustomOfflinePRC6SRC1100M The payment method name does not have the required format.

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC6SRC10000B The method does not have the required format

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC6SRC10001B The auxiliary1 value does not have the required format.

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_AUXILIARY1.

User Response:

CEPCustomOfflinePRC6SRC10002B The auxiliary2 value does not have the required format.

Severity:

Explanation: Return code pair - PRC_PARAMETER_FORMAT_ERROR, RC_CASSETTE_AUXILIARY2.

User Response:

CEPCustomOfflinePRC7SRC1099M The batch close time is not valid.

Severity:

Explanation: Return code pair - PRC_PARAMETER_VALUE_ERROR, RC_CASSETTE_BATCHCLOSETIME.

User Response:

CEPCustomOfflinePRC8SRC1100M The payment method name already exists.

Severity:

Explanation: Return code pair - PRC_DUPLICATE_OBJECT, RC_CASSETTE_METHOD.

User Response:

CEPCustomOfflinePRC14SRC23001M The specified order was not found.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_SELECT_ORDER_MISSING.

User Response:

CEPCustomOfflinePRC14SRC23002M An SQL exception occurred during a query for orders.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_SELECT_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC23003M An SQL exception occurred while closing an order query session.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_SELECT_CLOSE_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC23005M An SQL exception occurred while creating the order in the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_CREATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC23007M An SQL exception occurred while updating the order in the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_UPDATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC23009M An SQL exception occurred while deleting the order from the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_ORDER_DELETE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC24001M The specified payment was not found.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_SELECT_PAYMENT_MISSING.

User Response:

CEPCustomOfflinePRC14SRC24002M An SQL exception occurred during a query for Payments.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_SELECT_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC24003M An SQL exception occurred while closing the payment query session.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_SELECT_CLOSE_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC24005M An SQL exception occurred while creating the payment in the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_CREATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC24007M An SQL exception occurred while updating the payment in the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_UPDATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC24009M An SQL exception occurred while deleting the payment from the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_PAYMENT_DELETE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC25001M The specified credit was not found.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_SELECT_CREDIT_MISSING.

User Response:

CEPCustomOfflinePRC14SRC25002M An SQL exception occurred during a query for credits.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_SELECT_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC25003M An SQL exception occurred while closing a credit query session.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_SELECT_CLOSE_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC25005M An SQL exception occurred while creating the credit in the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_CREATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC25007M An SQL exception occurred while updating the credit in the datbase.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_UPDATE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC14SRC25009M An SQL exception occurred while deleting the credit from the database.

Severity:

Explanation: Return code pair - PRC_DATABASE_ERROR, RC_CREDIT_DELETE_SQL_FAILURE.

User Response:

CEPCustomOfflinePRC55SRC0M The receive payment command is not supported.

Severity:

Explanation: Return code pair - PRC_COMMAND_NOT_SUPPORTED, RC_NONE.

User Response:

Appendix C. National Language Support (NLS) information for WebSphere Commerce Payments

This appendix provides supplemental information on National Language Support (NLS) for Websphere WebSphere Commerce Payments. It is important that you review this information if you are installing a National Language Version (NLV) of the WebSphere Commerce Payments.

NLS hints and tips

Adobe Acrobat Reader limitation
PDF files may not display correctly for some languages on the AIX platform. If you are unable to view or print WebSphere Commerce Payments PDF files (that is, the documentation files), you can view and print the files from a browser on Windows NT or Windows 2000. This is a limitation of the AIX Adobe Acrobat Reader.

Prerequisite products installed for the Dutch language
The prerequisite products that are shipped with the WebSphere Commerce Payments (that is, WebSphere Application Server, DB2 UDB, and IBM HTTP Server) are not translated into the Dutch language. The WebSphere Commerce Payments install program installs the English version of these products as required.

Code pages
If characters are not displayed correctly in Spanish, ensure that you are using code page ISO-8859-1.

Log directory
Do not use DBCS characters, or other special characters, when specifying the directory where log information will be stored.

WebSphere Commerce Payments user names
WebSphere Commerce Payments supports only those user names made up of characters in the following table. User names containing characters outside this set are not accessible. The WebSphere Commerce Payments PSOS400Realm on iSeries systems allows only valid OS/400 user names.

This table lists the supported character set for assigning WebSphere Commerce Payments user names.

Table 30. Portable Character Set (PCS) for WebSphere Commerce Payments User Names and others

ASCII Hex value	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
Character	space	!	″	#	$	%	&	’	(	)	*	+	,	_	.	/
ASCII Hex value	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
Character	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
ASCII Hex value	40	41	42	43	44	45	46	47	48	49	4A	4B	4 C	4D	4E	4F
Character	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
ASCII Hex value	50	51	52	53	54	55	56	57	58	59	5A	5B	5 C	5D	5E	5F

Table 30. Portable Character Set (PCS) for WebSphere Commerce Payments User Names and others (continued)

Character	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
ASCII Hex value	60	61	62	63	64	65	66	67	68	69	6A	6B	6 C	6D	6E	6F
Character	’	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
ASCII Hex value	70	71	72	73	74	75	76	77	78	79	7A	7B	7 C	7D	7E	
Character	p	q	r	s	t	u	v	w	x	y	z	{	l	}	~	

NLS hints and tips for iSeries

The following table outlines the recommended system values for the languages in which WebSphere Commerce Payments is available.

Language Feature Code	**QCCSID**	**QCNTRYID**	**QLANGID**
2987 Traditional Chinese	937	TW	CHT
2989 Simplified Chinese	935	CN	CHS
2924 English	37	US	ENU
2938 English Uppercase DBCS	5026 or 5035	US	ENP
2950 English Uppercase	37	GB	ENP
2984 English Upper/Lower DBCS	5026 or 5035	US	ENU
2928 French	297	FR	FRA
2940 French (Swiss)	500	CH	FRS
2966 French (Belgium)	500	BE	FRB
2981 French (Canada)	500	CA	FRC
2929 German	273	DE	DEU
2939 German (Swiss)	500	CH	DES
2932 Italian	280	IT	ITA
2942 Italian (Swiss)	1144	CH	ITS
2962 Japanese	5026 or 5035	JP	JPN
2986 Korean	933	KR	KOR
2980 Brazilian Portuguese	37	BR	PTB
2931 Spanish	284	ES	ESP

Note: A WebSphere Commerce Payments instance may not be created with CCSID 5026 (CRTPYMMGR CCSID (5026)), but it may run on a 5026 system.

NLS hints and tips for AIX

The following table lists each language and the corresponding locale that is supported by WebSphere Commerce Payments on AIX.

Table 31. Supported Languages and Locales on WebSphere Commerce Payments for AIX

Language	**Locale**
Brazilian Portuguese	pt_BR
Dutch	nl_NL
English	en_US
French	fr_FR
German	de_DE
Italian	it_IT
Japanese	Ja_JP
Korean	ko_KR
Simplified Chinese	zh_CN
Spanish	es_ES
Traditional Chinese	Zh_TW

Appendix D. Troubleshooting problems and Performance tuning

Troubleshooting

Each business model possesses unique requirements and, as such, utilizes the WebSphere Commerce Payments functions somewhat differently. Often, the WebSphere Commerce Payments can be tuned to maximize its performance and functionality for a given business environment.

Tuning for high-performance environments

Wait time for TCP/IP sockets

Each request to the WebSphere Commerce Payments results in a TCP socket going into the `TIME_WAIT` state and remaining there for several minutes. For machines that service a high volume of requests, there may be a large number of sockets in the `TIME_WAIT` or `TIME_CLOSED` state, resulting in rejected requests (that is, a return code of `Cannot connect to WebSphere Commerce Payments`). This behavior is expected (and necessary) for all TCP connections.

TCP sockets move into the `TIME_WAIT` state for a period of time to ensure that any subsequent communication on the socket is not mistaken for new communication on a newly-bound socket. This period of time is, theoretically, 2 MSL (that is, twice the maximum segment lifetime). In practice, the `TIME_WAIT` default is four minutes on Windows and Solaris and two minutes on AIX. The default is approximately two minutes on iSeries systems. By altering the `TIME_WAIT` values on your operating system, high-volume users can reduce this problem. Following are examples of how the interval can be reconfigured on the Windows, Solaris, and AIX platforms.

Note: If an alternative stack is being used, other measures may be required.

Windows:

1. Locate in the Registry:

   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters
   \TcpTimeWaitDelay
   ```

   Note: If this entry does not exist in your Windows Registry, you must create it, following step 2 below.
2. Edit this entry as a new `DWORD` item.
3. Set it to any value between 30 and 300 (value represents the number of seconds). It is recommended that you set this value to 30.

AIX:

```
no -o tcp_timewait=1
```

Note: The value (1) is in 15 second increments. For example, 1 equates to 15 seconds, 2 equates to 30 seconds, and so on. It is recommended that you set the value to 1 or 2.

Solaris:

```
ndd -set /dev/tcp tcp_close_wait_interval 30000
```

Note: The value (30000) is in milliseconds.

Performance tuning parameters

WebSphere Commerce Payments Version 3.1.1 contains tuning parameters that allow you to control WebSphere Commerce Payments internal resource allocation.

Note: These parameters should only be modified by an administrator who is very familiar with the WebSphere Commerce Payments. Be aware that setting either of these parameter values too high could cause performance degradation or an outright failure of the WebSphere Commerce Payments at startup time.

It is *highly recommended* that you:

- make changes in small increments only.
- change only one parameter at a time; then measure the effect before making another change.
- thoroughly test changes on a non-critical system before using them on a production system.

The WebSphere Commerce Payments manages the following sets of thread pools:

Protocol thread pool
The threads in this pool are assigned the task of processing *server-side* protocol-specific messages within a cassette. The default number of threads in this pool is 8, and there is one such pool for each cassette that defines its own ComPoints. The property, `wpm.ppoolsize`, changes the Protocol thread pool size. The pool size can be changed by completing the following steps in the WebSphere Application Server administrative console.

1. Open the WebSphere Application Server Administrative console.
2. Expand **WebSphere Administrative Domain**.
3. Expand **Nodes**.
4. Expand the host name for the system where WebSphere Commerce Payments is installed.
5. Click **WebSphere WebSphere Commerce Payments**.
6. Select the **JVM Settings** tab page. In the **System Properties** box, add the parameter **wpm.ppoolsize**. If this value is already specified, change the parameter value of the protocol thread pool size.
7. Add the parameter value `com.ibm.commerce.payment.realm.WCSRealm`.
8. Click **Apply**.
9. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

Given the nature of the processing that is typically involved with server-side protocol messages (increased levels of network communication), these threads are typically held for a long time for each given request. Therefore, a cassette that relies on protocol threads to process protocol messages should be configured with a large enough protocol thread pool to accommodate a typical number of concurrent transactions involving the significant protocol messages.

Service thread pool
The threads in this pool are used by cassettes as well as the framework to perform current or future tasks in the background. The default number of threads in this pool is 3. The property `wpm.spoolsize` changes the Service

thread pool size. For release 3.1.1, you can change the pool size by completing the following steps in the WebSphere Application Server administrative console.

1. To specify the service thread pool, complete steps 1–5 in the previous **Protocol thread pool** section.
2. Select the **JVM Settings** tab page. In the **System Properties** box, add the parameter **wpm.spoolsize**. If this value is already specified, change the parameter value of the protocol thread pool size.
3. Select **Apply** to update the settings.
4. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

Initially, the most common uses for service threads was to process tasks that were scheduled to execute at some point in the future. Examples of such tasks are retried transactions with the cassette's ôback endôprocessor, as well as periodic maintenance tasks such as the SET cassette's periodic exchange of PCerts with the gateway.

wpm.disableDuplicateOrderCheck

This parameter instructs WebSphere Commerce Payments to bypass duplicate order checking when processing a new order and eliminate access to the database. This results in higher transaction throughput. It is recommended to use this parameter only if your merchants will not be generating duplicate order numbers. Doing so causes these orders to fail. To enable this parameter, complete the following steps through the WebSphere Application Server administrative console:

1. To specify the wpm.disableDuplicateOrderCheck parameter, complete steps 1–5 described in the previously mentioned **Protocol thread pool** section.
2. Select the **JVM Settings** tab page. In the **System Properties** box, add the parameter **wpm.spoolsize**. If this value is already specified, change the parameter value of the protocol thread pool size.
3. Add the parameter value **1**.
4. Select **Apply** to update the settings.
5. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

You can also use the WebSphere Application Server administrative console to set the number of concurrent connections from the WebSphere Commerce Payments to the database (WPM Data Source). After bringing up the console window, complete the following:

1. Select the topology view and expand the following to specify the pool size values:
   a. Expand **Resources**.
   b. Expand **JDBC Providers**.
   c. Expand **WPM DB Driver**.
   d. Select the **Data Sources** folder.
   e. Select the **Connection Pooling** tab from the property pane.
2. Specify the new value for the **Maximum pool size** parameter.
3. Select **Apply** to update the settings.

Note: Be aware that your database product may impose some limit on the number of concurrent connections that you can establish to your database at a given time. This may either be a technical limit or one imposed by the license agreement under which you purchased the database product. These factors must be considered while adjusting the above parameters. If you exceed such a limit by setting either of these parameters too high, the Payment Servlet initialization will fail.

Managing journal receivers on iSeries

As new orders and payments are processed by WebSphere Commerce Payments, the Payments instance database collection will grow in size. If the iSeries storage space begins to fill up, the system operator may receive warning messages such as `CPI099C - Critical storage lower limit reached`. If that happens you should consider freeing up space in the WebSphere Commerce Payments instance database.

The WebSphere Commerce Payments instance database collection is created with journaling set to *YES. This means that the instance library will contain database journal receivers (entries that indicate when events such as changes to a database file occur). In iSeries, the journal QSQJRN is built in the WebSphere Commerce Payments instance library, and journal receivers named QSQJRNXXXX are created in the instance library. Old journal receivers in the instance library (named QSJRNXXXX with type *JRNRCV) may be deleted to free up space if the WebSphere Commerce Payments instance library is getting too large. The most recent journal receiver will still be attached to the database and should not be deleted.

For more information about cleaning up storage, refer to the *AS/400 System Operation* manual (SC41-4203).

Error and trace logs

As you start to use WebSphere Commerce Payments, you may need to diagnose problems that you encounter while actually using the application. Error and trace logs are available to support this type of problem diagnosis. For more details on Error and trace logs, refer to the WebSphere Commerce Payments *Installation Guide*.

pmError
Errors associated with the Payment Servlet are recorded in the pmError file located in the WebSphere Commerce Payments `log` directory.

pmTrace1.log and pmTrace2.log
Support for these logs can be enabled through the WebSphere Commerce Payments user interface Trace settings. You can specify the level and size of the trace logs through the WebSphere Commerce Payments user interface. In addition, you can view the trace log files by executing the FormatTrace command, which invokes a utility program that formats the information into a readable text file.

Windows and UNIX systems

Diagnosing a problem could include accessing your error log or your trace logs. Both the error and trace logs are written to the Trace Log Path on the Trace window of the WebSphere Commerce Payments user interface. The error log, called `PMError`, cannot be disabled.

Tracing to the trace logs, called `PMTrace1.log` and `PMTrace2.log`, can be enabled through the WebSphere Commerce Payments user interface Trace Settings. Additionally, the level of tracing and the size of the trace logs can be configured through the WebSphere Commerce Payments user interface. Trace is fully configurable while the WebSphere Commerce Payments is running.

The WebSphere Commerce Payments provides the *FormatTrace* (`FormatTrace.cmd` on Windows platforms) utility program which allows the trace logs to be formatted into a readable text file. FormatTrace is used to format traces from the WebSphere Commerce Payments (`PMTrace1.log` and `PMTrace2.log`). The *FormatTrace* utility is identified and invoked in the directory from which the WebSphere Commerce Payments is started. For example:

```
FormatTrace <<logpath>> <<outputFileName>>
```

where `<<logpath>>` is the fully-qualified name of the directory containing the unformatted trace files, and `<<outputFileName>>` is the name of the file containing the readable text.

There is an additional file called `WebSphere.log` that may contain useful information for trouble analysis. This file is found in the \logs directory where WebSphere is installed.

iSeries systems

The WebSphere Commerce Payments error log, called PMError, cannot be disabled. The WebSphere Commerce Payments error log for the current invocation of the WebSphere Commerce Payments is viewable from the View Log menu item on the WebSphere Commerce Payments Tasks page. The error logs for previous invocations of the WebSphere Commerce Payments are available in the PMError.bak file.

Tracing to the WebSphere Commerce Payments trace logs, called PMTrace1.log and PMTrace2.log, can be enabled through the WebSphere Commerce Payments user interface Trace settings. The trace logs can be formatted using the Dump Payment Trace (DMPPYMTRC) CL command. Issuing the following command will format the WebSphere Commerce Payments trace into an IFS file for the WebSphere Commerce Payments instance called PYMINST:

```
DMPPYMTRC PYMMGR(PYMINST) TYPE(*PYMMGR) TOFILE('/tmp/payment_trace.txt')
```

Another way to format traces or view error logs is through the Service menu item on the WebSphere Commerce Payments Tasks page.

Error messages

For each message listed here, the following information is presented:

Severity

Information
No user action is required.

Warning
You should check to determine whether you need to change anything.

Error The specific process or application needs to be fixed and restarted.

Severe
The WebSphere Commerce Payments must be restarted after resolving the problem.

Explanation
The condition that was detected.

User Response
A suggested action or list of steps to remedy the problem.

Notes:

1. Protocol-specific (that is, SET or CyberCash) error messages are documented in the following cassette supplements for:
   - The *WebSphere Commerce Payments for SET Supplement*
   - The *WebSphere Commerce Payments for CyberCash Supplement*
   - The *WebSphere Commerce Payments Cassette for VisaNet Supplement*
   - The *WebSphere Commerce Payments for BankServACH Supplement*
2. Exception messages containing ″too many files open″ indicate that you need to use the `ulimit –u` *nnn* command to increase the number of available file handles.

CEPFW0000 IBM WebSphere Commerce Payments ***version*****.**

Severity: Error

Explanation: This is used to display the Framework version number in the error log.

User Response: None

CEPFW0100 Error: Unrecognized Payment API call ***call_name*** **received by WebSphere Commerce Payments.**

Severity: Error

Explanation: A Payment API command was received whose command name is not recognized by IBM WebSphere Commerce Payments.

User Response: If the command was sent to the Payment API port by a user application, then correct this application to send an API command recognized by WebSphere Commerce Payments.

CEPFW0101 An SQL exception occurred while retrieving an order for Merchant ***merchant_number*****, order number** ***order_number*****:** ***text***

Severity: Error

Explanation: An SQL exception occurred while IBM WebSphere Commerce Payments attempted to reference the order table in the WebSphere Commerce Payments Configuration Database.

User Response: Test the connection to the database and list the table ETORDER to make certain that it has not been corrupted.

CEPFW0102 The ***flag*** **flag value of** ***flag_value*** **is not a supported value. Values must be –1, 0, or 1.**

Severity: Error

Explanation: The approve or deposit flag specified on a RECEIVEPAYMENT or APPROVE API command is not a supported value.

User Response: Reissue the command with the correct value parameter.

CEPFW0103 The required parameter ***parameter*** **was not specified on a** ***command_name*** **command.**

Severity: Error

Explanation: A required parameter was not provided in the specified command.

User Response: Reissue the command with the required parameter.

CEPFW0104 Parameter ***parameter*** **had length** ***length*** **which is less than the minimum length** ***length*****.**

Severity: Error

Explanation: The specified parameter has a minimum length.

User Response: Reissue the command with a valid parameter value.

CEPFW0105 The currency *currency_name* specified in a *command_name* command for Merchant *merchant_number*, order *order_number* does not match the currency *currency_name* specified on the corresponding RECEIVEPAYMENT command.

Severity: Error

Explanation: The CURRENCY parameter specified on the Payment API command must match the CURRENCY specified on the RECEIVEPAYMENT command for this order.

User Response: Reissue the command with the correct currency.

CEPFW0106 The merchant number *merchant_number* and payment system *payment_system* were not configured in the Payment System Configuration Table.

Severity: Error

Explanation: The Merchant specified has not been configured to use the specified payment system.

User Response: Change the PAYMENTSYSTEM parameter on the RECEIVEPAYMENT call to use a configured payment system.

CEPFW0107 The value *specified_exponent* specified for the AMOUNTEXP10 parameter does not match the expected value *exponent* for currency code *currency* as specified in the ISO 4217 currency list.

Severity: Error

Explanation: When specifying a known ISO 4217 currency, the AMOUNTEXP10 parameter must match the known exponent.

User Response: Reissue the command with the correct AMOUNTEXP10 (or omit it altogether).

CEPFW0108 Parameter *parameter* had a length of *length*, which exceeds the maximum allowable length *maxlength*.

Severity: Error

Explanation: The length of a command parameter exceeded the maximum number of bytes.

User Response: Reissue the command with a valid parameter value.

CEPFW0109 Parameter *parameter* must be numeric. The non-numeric value *value* was specified.

Severity: Error

Explanation: The specified parameter must contain a numeric value.

User Response: Reissue the command with a valid parameter value.

CEPFW0110 Parameter *parameter* must not be negative. The specified value *value* is less than zero.

Severity: Error

Explanation: The specified parameter must contain a numeric value greater than 0.

User Response: Reissue the command with a valid parameter value.

CEPFW0111 Parameter *parameter* is out of range. The specified value *value* is greater than the maximum of *maximum*.

Severity: Error

Explanation: The specified parameter must contain a numeric value no larger than the indicated maximum.

User Response: Reissue the command with a valid parameter value.

CEPFW0112 The value specified for CURRENCY, *value* , does not contain *length* bytes.

Severity: Error

Explanation: The value specified for the CURRENCY parameter does not contain the number of digits expected in a currency code.

User Response: Reissue the command with a valid CURRENCY value.

CEPFW0113 The value specified for CURRENCY, *value*, on a *command_name* command is invalid.

Severity: Error

Explanation: The value specified for CURRENCY is not a recognized country/region code according to the ISO 4217 standard. IBM WebSphere Commerce Payments expects the numeric string values of the currency code; for example, 840 rather than USD for U.S. dollars.

User Response: Reissue the command with a valid CURRENCY value.

CEPFW0123 The AMOUNTEXP10 *value* specified in a *command* command for Merchant *merchant_number*, order *order_number* does not match the AMOUNTEXP10 *value* specified on the corresponding RECEIVEPAYMENT command.

Severity: Error

Explanation: The AMOUNTEXP10 specified on the Payment API command must match the AMOUNTEXP10 specified on the RECEIVEPAYMENT command for this order.

User Response: Reissue the command with a matching AMOUNTEXP10 value.

CEPFW0126 The amount specified (*amount*) in a *command_name* command is not valid.

Severity: Error

Explanation: The amount specified is either negative, which is never valid or is 0 and the Payment API command is not APPROVEREVERSAL. The amount can be 0 only if the command is APPROVEREVERSAL or DEPOSITREVERSAL.

User Response: Reissue the command with a valid Amount value.

CEPFW0127 The CONTENTTYPECHARSET parameter specified (*value*) on a RECEIVEPAYMENT command is not a valid CONTENTTYPECHARSET string.

Severity: Error

Explanation: The Content Type and Charset are the two parts of the CONTENTTYPECHARSET string. The Content Type must be of the form type/sub-type. The values for the type and sub-type are defined in RFC 1521. The charset is optional in the specification of this parameter. It must be separated from the Content Type by a ";". RFC 1521 has the list of the international standards for the charset. The actual value is not checked by the WebSphere Commerce Payments.

User Response: Correct the value specified for the CONTENTTYPECHARSET.

CEPFW0148 The BATCHNUMBER *account_number*, specified in a *command_name* command, must be greater than zero.

Severity: Error

Explanation: The BATCHNUMBER on a Payment API command must be greater than zero.

User Response: Reissue the command with a valid BATCHNUMBER parameter.

CEPFW0150 A *command* command was sent for Merchant *merchant_number*/order *order_number*, which is not associated with an account number.

Severity: Error

Explanation: The order specified in the API command has not been associated with an account number.

User Response: Associate the order with an account number and reissue the command.

CEPFW0151 A *API* command was sent for a payment that does not exist.

Severity: Error

Explanation: A request has been made to perform an API command on a payment that does not exist.

User Response: Create a payment via the *APPROVE* command.

CEPFW0152 A *API* command was sent for a credit that does not exist.

Severity: Error

Explanation: A request has been made to perform an API command on a credit that does not exist.

User Response: Create a credit via the *REFUND* command.

CEPFW0153 A retrieve of batch *batch_number* for a BATCHCLOSE API command failed for Merchant *merchant_number*.

Severity: Error

Explanation: A *BATCHCLOSE* API command was issued, but no batch was found matching the BATCHNUMBER and MERCHANTNUMBER specified.

User Response: Verify that you have entered the correct values for the BATCHNUMBER and MERCHANTNUMBER API values.

CEPFW0154 A retrieve of account *account_number* for a BATCHOPEN, BATCHCLOSE or DELETEBATCH API command failed for Merchant *merchant_number*.

Severity: Error

Explanation: A *BATCHOPEN* or *BATCHCLOSE* or *DELETEBATCH* API command was issued, but no account was found matching the specified *MERCHANTNUMBER*.

User Response: Check that you have entered the correct *MERCHANTNUMBER* and *ACCOUNTNUMBER* for the API.

CEPFW0155 A retrieve of merchant *merchant_number* **for a BATCHOPEN, BATCHCLOSE or DELETEBATCH API command failed.**

Severity: Error

Explanation: A *BATCHOPEN* or *BATCHCLOSE* or *DELETEBATCH* API command was issued, but no merchant was found matching the merchant number.

User Response: Check that you have entered the correct merchant number for the API.

CEPFW0156 A retrieve of cassette *cassette_number* **for an administration API command failed.**

Severity: Error

Explanation: A administration API command was issued, but no cassette was found matching the cassette number.

User Response: Check that you have entered the correct cassette number for the API.

CEPFW0157 A retrieve of merchant *merchant_number* **for an administration API command failed.**

Severity: Error

Explanation: An administration API command was issued, but no merchant was found matching the merchant number.

User Response: Check that you have entered the correct merchant number for the API.

CEPFW0158 A retrieve of merchant *merchant_number* **for an API command operated on Orders failed.**

Severity: Error

Explanation: An API command operated on Orders was issued, but no merchant was found matching the merchant number.

User Response: Check that you have entered the correct merchant number for the API.

CEPFW0159 A retrieve of account *account_number* **for an API command operated on Orders failed.**

Severity: Error

Explanation: An API command operated on Orders was issued, but no account was found matching the account number.

User Response: Check that you have entered the correct account number for the API.

CEPFW0160 Parameter *parameter* **contains leading zeros (***value***), which are not valid.**

Severity: Error

Explanation: The specified parameter contains leading zeros, which are not valid.

User Response: Reissue the command with a valid parameter value.

CEPFW0162 The AMOUNT parameter value *Amount_number* **specified on a** *command_name* **command exceeds the maximum value allowed.**

Severity: Error

Explanation: The AMOUNT parameter on a Payment API command must be less than or equal to 2147483647.

User Response: Correct the amount specified on the command.

CEPFW0163 Parameter value *parameter* **was not valid.**

Severity: Error

Explanation: The specified parameter on a Payment API command must be number and between 0 and 2147483647.

User Response: Correct the value specified on the command for the parameter.

CEPFW0202 An SQL exception occurred while creating an object *object_key* **for Merchant** *merchant number* **:** *text*

Severity: Error

Explanation: An SQL exception occurred while attempting to create an object int the database. This could be due to an error connecting or writing to the database or due to an error in the data being stored.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPFW0203 An SQL exception occurred while updating an object *object_key* **for Merchant** *merchant number* **:** *text*

Severity: Error

Explanation: An SQL exception occurred while updating an object in the WebSphere Commerce Payments database. This could be due to an error connecting to or writing to the database or due to an error in the content of the data being stored.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPFW0204 An SQL exception occurred while reading an object *order_number* **from the database for Merchant** *merchant number***:** *text*

Severity: Error

Explanation: An SQL exception occurred while retrieving a record from the WebSphere Commerce Payments database. This could be due to an error connecting to or reading from the database, or due to an error in the content of the data that was read from the database.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPFW0205 An SQL exception occurred while accessing the database: *text***.**

Severity: Error

Explanation: An SQL exception occurred while accessing the WebSphere Commerce Payments database. This could be due to an error connecting to or accessing the database, or due to an error in the content of the data.

User Response: Check the connection to the database to make sure that there is not a problem with the communication between the WebSphere Commerce Payments machine and the database server.

CEPFW0206 Unable to locate a transaction for *name* **payment system after a message was received from the Cardholder.**

Severity: Warning

Explanation: A protocol message was received from a cardholder; no matching order was found. This could be due to one of the following reasons.

- WebSphere Commerce Payments found no record for this purchase. There will be a CEP0339 message logged as well.
- The payment system was not able to decode this message. There will be a CEP0338 message logged when this occurs.
- The type of the message received was not expected. Messages CEP0337 and CEP0338 will also be logged when this occurs.

User Response: If there is a related message logged for this error, check to see if there are any instructions for how to correct the error. Otherwise, contact your service representative.

CEPFW0207 An exception occurred while processing a message from the Payment Gateway: *hostname*

Severity: Error

Explanation: A Java exception occurred while processing a message received from a Payment Gateway. This prevented the protocol message processing from completing. There are many conditions that can cause this. Some of WebSphere Commerce Payments generated exceptions that cause this are:

- SET Transaction was not found; the transaction was not stored in the SET Message database. CEPFW0336 is also logged when this occurs.
- A SET authorize (approve) or capture (deposit) command was received and there is no transaction stored in the SET Message database for it. CEPFW0335 is also logged when this occurs.
- SQL exception occurred while logging SET Message record to the SET Message database. CEPFW0318 is also logged when this occurs.

User Response: See if there is a related message logged for this error; follow the instructions for those messages, if there were any. Otherwise, contact your service representative.

CEPFW0208 Unable to locate the merchant for *merchant_number* **after a message was received from the Cardholder.**

Severity: Warning

Explanation: A protocol message was received from a cardholder; no matching merchant was found.

User Response: Contact your service representative.

CEPFW0209 Unsupported Payment Type found in a payment system profile: *type***.**

Severity: Error

Explanation: A value was specified in the PAYMENTTYPE field of a payment system profile record that is not supported by the WebSphere Commerce Payments. This payment system is not recognized on Payment API commands to the WebSphere Commerce Payments.

User Response: Delete or correct the database record in the PaymentSystem database record that contains the unsupported Payment Type.

CEPFW0301 Read issued by Merchant *merchant_name* **to Acquirer** *acquirer_name* **failed:** *text*

Severity: Error

Explanation: An I/O exception occurred while reading from the Payment Gateway on the specified Acquirer.

User Response: Verify that the communication between the WebSphere Commerce Payments and the Acquirer is valid. Look for a CEPFW0606 message that corresponds to this. If there is a corresponding message, contact your service representative.

CEPFW0344 Retrieve of order failed for Merchant *merchant_number*, **order number** *order_number*: *text*.

Severity: Error

Explanation: An SQL exception occurred while retrieving an order from the Order table. This could be due to the order not being in the Order table or due to an error reading the record itself; it could also be due to a communication error between the WebSphere Commerce Payments and the database server.

User Response: Test the connection to the WebSphere Commerce Payments database, verify that this order is in the Order table, and then retry the operation that caused this error.

CEPFW0358 A null return code was returned by processToken while processing a command *command_name*.

Severity: Error

Explanation: There was an unexpected error while attempting to process this command.

User Response: This is an internal WebSphere Commerce Payments error. Contact your service representative.

CEPFW0412 There was an SQL Exception while reading a payment system profile record from *tablename*: *text*.

Severity: Error

Explanation: An SQL exception occurred while writing a payment system profile to the Payment System Configuration table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This is likely due to an error mismatch in the Payment System Configuration table definition and the one expected by the WebSphere Commerce Payments; the data type of one of the fields that was retrieved from the database row did not match what was expected.

User Response: Refer to the SQL state information to get specific details about the nature of the problem.

CEPFW0413 An SQL Exception occurred while reading a PaymentSystem Configuration record from *table_name*: *text*

Severity: Error

Explanation: An SQL exception occurred while retrieving the PaymentSystem Configuration table. The SQL exception text describes the exception and provides SQL state information that can be reference in the XOPEN SQL specification. This is likely due to an error or disruption in the communication between the WebSphere Commerce Payments and the database server.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Verify that the connection to the database server is valid.

CEPFW0414 Either the MERCHANTNAME *merchant_number* **or the PAYMENTSYSTEMNAME** *value* **value is null.**

Severity: Error

Explanation: Both of the MERCHANTNAME and PAYMENTSYSTEMNAME fields of the Payment System Configuration record must be set, since they are the SQL keys for the Payment System Configuration table.

User Response: Verify that both fields are filled in on the WebSphere Commerce Payments Configuration table. If they are and this error occurs, contact your service representative.

CEPFW0415 The MERCHANTNAME value, *value*, **should be a numeric string of one to nine digits.**

Severity: Error

Explanation: A value was specified for the MERCHANTNAME field of an Acquirer Profile that is not a numeric string. The MERCHANTNAME field is a 1 to 9 digit numeric string.

User Response: Correct the value of MERCHANTNAME in the Payment System Configuration table.

CEPFW0428 Unable to load JDBC driver *driver_name*: **driver name is not valid or IBM WebSphere Commerce Payments environment is not configured properly.**

Severity: Error

Explanation: The JDBC driver name specified was not found in the WebSphere Commerce Payments environment. The JDBC driver may have been typed incorrectly or the CLASSPATH environment variable may not have been changed to specify the location of the JDBC driver in the WebSphere Commerce Payments environment.

User Response: Ensure JDBC driver name is typed correctly and CLASSPATH specifies the location of the JDBC driver.

CEPFW0429 The DBDRIVER command line option has not been set.

Severity: Error

Explanation: The JDBC driver name was not specified on the DBDriver parameter in the PaymentServlet init parameters.

User Response: Ensure JDBC driver name is specified on the DBDRIVER parameter.

CEPFW0430 Failed to access class ***class_name*****.**

Severity: Error

Explanation: The JDBC driver class is not specified in the CLASSPATH environment variable.

User Response: Ensure that the CLASSPATH contains the path to the JDBC Driver class.

CEPFW0431 There was an attempt to instantiate the interface or abstract class ***class_name***

Severity: Error

Explanation: Attempting to instantiate a JDBC Driver that is an interface or abstract class.

User Response: Ensure that the JDBC driver being used is not an interface or abstract class. If it is not an interface or abstract class, contact your service representative.

CEPFW0435 IBM WebSphere Commerce Payments JDBC database URL was not initialized.

Severity: Error

Explanation: The JDBC URL was not initialized.

User Response: Consult the database vendor documentation to determine the proper JDBC URL syntax.

CEPFW0436 The IBM WebSphere Commerce Payments database owner was not specified.

Severity: Error

Explanation: The WebSphere Commerce Payments database owner was not set on the DBOWNER parameter in the PaymentServlet init parameters.

User Response: Provide the WebSphere Commerce Payments database owner on the DBOWNER parameter.

CEPFW0437 IBM WebSphere Commerce Payments database user ID was not specified.

Severity: Warning

Explanation: The WebSphere Commerce Payments database userid was not initialized in the PaymentServlet init parameters..

User Response: Provide the WebSphere Commerce Payments database user ID on the DBUSERID parameter.

CEPFW0438 IBM WebSphere Commerce Payments ***which*** **password was not specified.**

Severity: Warning

Explanation: The WebSphere Commerce Payments database or root password was not initialized in the PaymentServlet init parameters

User Response: Provide the WebSphere Commerce Payments required password in the PaymentServlet init parameters.

CEPFW0459 Failed to connect to the database.

Severity: Error

Explanation: This error may be caused by:

- Specification of wrong URL address of the database
- Specification of the wrong login
- Specification of wrong password
- Specification of the wrong JDBC driver

User Response: Supply the correct login, password, JDBC driver, and URL address for the database.

CEPFW0460 Failed to execute the following SQL Statement *sql_query.*

Severity: Error

Explanation: This error may be caused by:

- Mismatch in the number of the fields of the table
- Mismatch in the data type
- The specified field in that table does not exist

User Response: Verify that the database field definitions match what is required by the WebSphere Commerce Payments.

CEPFW0461 Failed to close the database connection.

Severity: Error

Explanation: This error may have occurred because the database connection no longer exists.

User Response: The database may have been closed or shut down before the WebSphere Commerce Payments connection was closed. Start the database and WebSphere Commerce Payments again.

CEPFW0462 Failed to close the SQL prepare statement.

Severity: Error

Explanation: An SQL exception has occurred while closing a prepare statement since the prepare statement was closed previously.

User Response: The connection with the database has been lost before closing the SQL prepare statement. Verify that the database connection exists and the database is running. Start the database and WebSphere Commerce Payments.

CEPFW0465 Failed to access the database product name successfully.

Severity: Error

Explanation: An error may have occurred while creating the database's metadata.

User Response: The database URL may not exist or is invalid. Also, the connection to the database may not exist. Verify that the URL is valid and the connection to the database exists. If necessary, start the database and WebSphere Commerce Payments.

CEPFW0467 Failure attempting to add, update, or delete a Cassette Profile with company name *name* and payment system name *name*.

Severity: Error

Explanation: IBM WebSphere Commerce Payments could not successfully perform an SQL command against the Cassette Configuration table, due to missing information.

User Response: Provide the required information and retry the command.

CEPFW0468 No cassettes are configured in the Cassette Configuration table. IBM WebSphere Commerce Payments framework must load at least one cassette in order to process payments.

Severity: Error

Explanation: Payment cassettes add an entry to the Cassette Configuration table when they are installed. No entry has been found in this table so no cassettes have been configured.

User Response: Restore the SET cassette configuration or install another payment cassette.

CEPFW0469 An SQL exception occurred while reading *reading_number*: *text*

Severity: Error

Explanation: SQL exception occurred while reading a WebSphere Commerce Payments database table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPFW0470 SQL exception executing a statement *statement_name* on *x_item*: *text*

Severity: Error

Explanation: SQL exception occurred while executing an SQL command on a WebSphere Commerce Payments database table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPFW0471 An SQL exception occurred while preparing an SQL statement for *x_item*: *text*

Severity: Error

Explanation: SQL exception occurred while constructing an SQL statement for a WebSphere Commerce Payments database table. The SQL exception text describes the exception and provides SQL state information that can be looked up in the XOPEN SQL specification. This could be the result of a disruption in communication between the WebSphere Commerce Payments and the database server, or a discrepancy between the table definition and the definition expected by the WebSphere Commerce Payments.

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPFW0472 A command was issued for which there was no associated object of type *type*.

Severity: Error

Explanation: A required object of the specified type for the command was not found. This problem could be the result of parameter error or an undefined object.

User Response: Ensure that an object corresponding to the identifiers presented does exist.

CEPFW0473 A command was issued that tried to operate on an inactive object of type *type*.

Severity: Error

Explanation: The specified command requires certain administration objects to be active. An object of the specified type is inactive, resulting in a failed operation.

User Response: Ensure that all relevant objects are active by enabling them through the API or User Interface, and retry the command.

CEPFW0500 An API command *command_name* was sent to create the *object_name* object that already exists.

Severity: Error

Explanation: A request has been made to perform an API command, which try to create an object. However, the object already exists in the system.

User Response: Change the parameters to create a different object of the same type. You can also issue a query to find out more information on this object.

CEPFW0501 An API command *command_name* was sent to *modify_or_delete* the *object_name* object that does not exist.

Severity: Error

Explanation: A request has been made to perform an API command to modify or delete an object. However, the object does not exist in the system.

User Response: You can issue a query to find out which objects are available in the system.

CEPFW0502 An API command was sent to operate on the *object_name* object. However, the required *object_name* object does not exist.

Severity: Error

Explanation: A request has been made to perform an API command to the specified object. However, other required objects such as cassette or merchant, do not exist in the system.

User Response: You can issue a query to find out the status of those required objects.

CEPFW0503 The value *value* of *name* retrieved from the database for the *object_name* is not valid.

Severity: Error

Explanation: A value retrieved from the database has an invalid value. This could be due to the database tampered, or an internal error.

User Response: Ask your database administrator to check the database. Contact your service representative for further assistance.

CEPFW0504 The commit database operation failed.

Severity: Warning

Explanation: The commit operation failed. The WebSphere Commerce Payments will retry the commit operation.

User Response: You may not need to take any actions unless the retry fails.

CEPFW0606 An internal error occurred.

Severity: Error

Explanation: An internal error occurred in the WebSphere Commerce Payments.

User Response: If some required operation or service is not functioning properly, contact IBM Service.

CEPFW0612 Internal error occurred.

Severity: Error

Explanation: An internal error occurred in the WebSphere Commerce Payments.

User Response: If some required operation or service is not functioning properly, contact IBM Service.

CEPFW0617 An IOException occurred while opening a connection to host name *hostname*, port *port_number*.

Severity: Error

Explanation: An I/O exception occurred when the WebSphere Commerce Payments attempted to open a socket to the host name specified using the port number indicated. This could be because the port is already in use, because there was a network error, or because the host name indicated either is not valid or is not responding.

User Response: Test the connection to the host name specified. Verify that the specified host is still up and running.

CEPFW0619 The WebSphere Commerce Payments did not initialize because of the following exception: *text*.

Severity: Severe

Explanation: An error has occurred that has caused WebSphere Commerce Payments to not initialize.

User Response: Try to resolve the problem by using the exception data. Otherwise contact IBM Service.

CEPFW0620 The user *user_name* is not allowed to perform the operation requested.

Severity: Error

Explanation: The specified user attempted an operation for which the user is not authorized.

User Response: Contact a user with more authority than this user or give this user the authority to perform the operation.

CEPFW0621 The realm *realm_name* specified in the *file_name* file is not a valid WebSphere realm.

Severity: Severe

Explanation: The realm specified in the file determines the base set of users that may be granted access to WebSphere Commerce Payments functionality. Without a properly specified realm, the set of users cannot be determined.

User Response: Specify a valid WebSphere Application Server realm name in the file.

CEPFW0622 The database table *table_name* could not be read, probably because a database connection could not be obtained. Users' authorization rights may not take effect immediately. The exception information is: *text*

Severity: Error

Explanation:

User Response: Refer to the SQL state information to get specific details about the nature of the problem. Test the database server connection and verify that the table definition matches the definition expected by the WebSphere Commerce Payments.

CEPFW0701 IBM WebSphere Commerce Payments Exiting due to unhandled exception: *text*.

Severity: Severe

Explanation: An unhandled exception has occurred that has caused the WebSphere Commerce Payments to terminate unexpectedly.

User Response: Contact your service representative.

CEPFW0702 IBM WebSphere Commerce Payments has started successfully.

Severity: Information

Explanation: The WebSphere Commerce Payments has started successfully.

User Response: None

CEPFW0703 IBM WebSphere Commerce Payments has stopped.

Severity: Information

Explanation: The WebSphere Commerce Payments has stopped.

User Response: None

CEPFW0704 The directory *directory_name* is not valid.

Severity: Error

Explanation: WebSphere Commerce Payments does not have access to the specified path, or the path is not valid.

User Response: Check the existence of the directory and retry the operation.

CEPFW0705 The system does not have write permissions for directory *directory_name*.

Severity: Error

Explanation: WebSphere Commerce Payments does not have write permissions for the specified path.

User Response: Either change the permissions of the specified directory or select a valid one.

CEPFW0707 IBM WebSphere Commerce Payments could not start because no input parameters were provided.

Severity: Error

Explanation: The WebSphere Commerce Payments could not start due to missing parameters.

User Response: Start the WebSphere Commerce Payments using the following parameters:

```
DBDRIVER=JDBC driver class name
DBJDBCURL=jdbc:db:dbname
DBOWNER=dbuser
DBUSERID=dbuser
DBPASSWORD=dbuserpw
```

CEPFW0711 Exception occurred in the Java layer. Stack trace: *text*.

Severity: Error

Explanation: The Java layer has thrown an exception.

User Response: Use the stack trace and surrounding messages to resolve the error.

CEPFW0712 Order number *number* already exists for Merchant *merchant_number*.

Severity: Error

Explanation: The Merchant attempted to create an order using an existing order number.

User Response: Use a unique order number.

CEPFW0713 Payment number *number* already exists for Merchant *merchant_number* and order *number*.

Severity: Error

Explanation: The Merchant attempted to create a payment using an existing payment number.

User Response: Use a unique payment number.

CEPFW0714 Credit number *number* already exists for Merchant *merchant_number* and order *number*.

Severity: Error

Explanation: The Merchant attempted to create a credit using an existing credit number.

User Response: Use a unique credit number.

CEPFW0715 A batch with BATCHNUMBER *number* already exists for Merchant *merchant_number*.

Severity: Error

Explanation: The Merchant attempted to open a batch using an existing *BATCHNUMBER*.

User Response: Use a unique BATCHNUMBER for each batch.

CEPFW0716 WebSphere Commerce Payments has enabled load balancing with group name *group_name*, host name *host_name*, and server name *server_name*.

Severity: Information

Explanation: WebSphere Commerce Payments has registered with the Work Load Manager and Domain Name Server for the Sysplex with the group name, host name, and server name given in the message.

User Response: None

CEPFW0717 WebSphere Commerce Payments has enabled load balancing with NDHostname *NDHostname* and host name *host_name*.

Severity: Information

Explanation: WebSphere Commerce Payments has registered with the Work Load Manager and net.dispatcher for the Sysplex with the net.dispatcher host name, and host name given in the message.

User Response: None

CEPFW0719 The WebSphere Commerce Payments could not start because an unrecognized command line parameter was detected.

Severity: Error

Explanation: Unrecognized command line parameter detected.

User Response: Start the WebSphere Commerce Payments using the following parameters:

```
DBDriver=JDBC driver class name
DBjdbcURL=jdbc:db:dbname
DBOwner=dbuser
DBUserID=dbuser
DBPassword=dbuserpw
```

CEPFW0720 Cassette *cassette_name* **failed to start.**

Severity: Error

Explanation: The cassette failed to successfully execute one of its startup functions.

User Response: Look for a more specific error message from the cassette.

CEPFW0721 Cassette *cassette_name* **does not implement a required method.**

Severity: Error

Explanation: Cassettes must implement all of the IBM WebSphere Commerce Payments's Cassette Programming Interface. One of the methods was not found.

User Response: Check the exception text for the name of the missing method.

CEPFW0722 Cassette *cassette_name* **failed to load. Check your class path**

Severity: Error

Explanation: The Java runtime could not find the Cassette class.

User Response: Make sure your class path points to the zip file containing the Cassette or to the top of the tree where the Cassette class file can be found.

CEPFW0723 An instantiation exception occurred while attempting to load cassette *cassette_name***.**

Severity: Error

Explanation: An instance exception occurred while attempting to load the specified cassette.

User Response: Contact your service representative.

CEPFW0724 An illegal access exception occurred while attempting to load cassette *cassette_name***.**

Severity: Error

Explanation: An illegal access exception occurred while attempting to load the specified cassette.

User Response: Contact your service representative.

CEPFW0725 An illegal argument exception occurred while attempting to load Cassette *cassette_name***.**

Severity: Error

Explanation: An illegal argument exception occurred while attempting to load the specified cassette.

User Response: Contact your service representative.

CEPFW0726 Unable to register group name *group_name***, host name** *host_name***, and server name** *server_name* **with the Sysplex Domain Name Server/WLM.**

Severity: Error

Explanation: There was an error trying to register the WebSphere Commerce Payments with the Work Load Manager/Domain Name Server for the Sysplex. The WebSphere Commerce Payments continues to run, but will not take part in workload balancing.

User Response: Verify that there is an active Domain Name Server for the sysplex. Verify the values in the ETSYSPLEX table.

CEPFW0728 Unable to open log file *file_name* **for writing.**

Severity: Error

Explanation: There was an error while trying to open the log file for writing.

User Response: Verify that the log path specified in the WebSphere Commerce Payments profile is valid and that the WebSphere Commerce Payments has write permission.

CEPFW0731 The *CassetteName* **Cassette has stopped.**

Severity: Information

Explanation: The named cassette has stopped.

User Response: None

CEPFW0732 Cassette *cassette*

Severity: Information

Explanation: None

User Response: None

CEPFW0733 There are no Cassettes installed.

Severity: Information

Explanation: There are no payment cassettes currently entered in the cassette configuration table.

User Response: None

CEPFW0734 There are no currently active Cassettes.

Severity: Information

Explanation: There are no payment cassettes currently entered in the cassette configuration table, or all the cassettes have been stopped.

User Response: None

CEPFW0735 The following Cassettes are installed:

Severity: Information

Explanation: A list of the payment cassettes that have been entered in the cassette configuration table follows.

User Response: None

CEPFW0736 The following Cassettes are active:

Severity: Information

Explanation: A list of the payment cassettes that are currently able to process requests follows.

User Response: None

CEPFW0737 Unknown operator command *command***.**

Severity: Error

Explanation: The command entered at the operator console was not a valid WebSphere Commerce Payments command.

User Response: Enter a valid WebSphere Commerce Payments command at the operator console

CEPFW0738 The operator command *command* **completed with primary return code** *prc* **and secondary return code** *src***.**

Severity: Information

Explanation: The command entered at the operator console completed with the specified return codes.

User Response: If the primary and secondary return codes are not 0, look up the return codes in the WebSphere Commerce Payments Programmer Guide and Reference. Correct the operator command if necessary.

CEPFW0740 TRACESETTING must be equal to either ON or OFF.

Severity: Error

Explanation: The syntax of the operator command was not correct.

User Response: Correct the command syntax and re-enter the operator command.

CEPFW0756 The cassette *cassette_name* **is already active.**

Severity: Error

Explanation: The specified cassette is already accepting requests.

User Response: None

CEPFW0757 The cassette *cassette_name* **is not active.**

Severity: Error

Explanation: The specified cassette is not currently accepting requests.

User Response: None

CEPFW0758 The cassette *cassette_name* **is not installed.**

Severity: Error

Explanation: The specified cassette has not been installed.

User Response: If the cassette has not been installed, install it and add the cassette to the WebSphere Commerce Payments configuration database. If the cassette has been installed, add it to the WebSphere Commerce Payments configuration database.

CEPFW0761 An *exception* **exception occurred while processing the BECOME command.**

Severity: Error

Explanation: An error of the type specified in the message occurred when the BECOME operator command was issued.

User Response: Contact the WebSphere Commerce Payments administrator with the exception information.

CEPFW0762 The specified host already has an active WebSphere Commerce Payments.

Severity: Error

Explanation: The WebSphere Commerce Payments BECOME command is valid only for stopped WebSphere Commerce Paymentss.

User Response: Re-enter the command with the name of the stopped WebSphere Commerce Payments whose pending transactions should be restarted.

CEPFW0763 The BECOME command requires a WebSphere Commerce Payments host name.

Severity: Error

Explanation: The BECOME command was entered on the operator console, but no target name was entered.

User Response: Re-enter the command with the name of the host whose pending WebSphere Commerce Payments transactions should be restarted.

CEPFW0764 Error *error_type* occurred while the *cassette_name* cassette was processing a *api_command_name* API request.

Severity: Warning

Explanation: The cassette did not process the API request successfully. An API response will still be generated by the WebSphere Commerce Payments, but it will not contain the cassette-specific information from this cassette. The CEPFW0711 error message contains further information about the error.

User Response: Contact the service group for this cassette for resolution.

CEPFW0765 The WebSphere Commerce Payments DTD definition *dtd_file_name* was not found in the classpath *websphere_classpath*

Severity: Warning

Explanation: An API request was received without a dtdPath parameter. In response, the WebSphere Commerce Payments attempted to send the full WebSphere Commerce Payments DTD definition with the XML response file. However, the DTD definition could not be found. An XML response was generated with no reference to a DTD file.

User Response: Correct the WebSphere Application Server classpath so that it includes the WebSphere Commerce Payments DTD file, restart the Web server and try again.

CEPFW0766 The WebSphere Commerce Payments DTD definition *dtd_file* was found but could not be successfully parsed.

Severity: Warning

Explanation: An API request was received without a dtdPath parameter. In response the WebSphere Commerce Payments attempted to send the full WebSphere Commerce Payments DTD definition with the XML response file. However, the DTD definition file could not be successfully parsed. The CEPFW0711 error message contains further information about the error.

User Response: Correct the WebSphere Commerce Payments DTD file. Otherwise, contact your service representative.

CEPFW0810 An error occurred while attempting to *en_decrypt* the encryption key.

Severity: Error

Explanation: Unable to encrypt/decrypt the encryption key. The CEPFW0711 error message contains further information about the error.

User Response: Make sure that the root (database) password is correct. Otherwise, contact IBM service.

CEPFW0811 The encryption key type is not supported.

Severity: Error

Explanation: The encryption key type requested is not support.

User Response: Only request supported encryption key.

CEPFW0812 An error occurred while attempting to *validate_generate* the encryption key.

Severity: Error

Explanation: Unable to generate the encryption/decryption key. The CEPFW0711 error message contains further information about the error.

User Response: Contact IBM service.

CEPFW0813 The realm specified in the PaymentServlet init parameters (eTill.RealmClass) could not be loaded.

Severity: Error

Explanation: The eTill.RealmClass setting in the PaymentServlet init parameters identifies which plug-in realm is used by PaymentManager. Either this class could not be found in WebSphere Application Server's classpath, or the realm settings in the PaymentServlet init parameters are not valid.

User Response: Set eTill.RealmTraceFlag=1 in the PaymentServlet init parameters to enable realm tracing and use FormatTrace to see if additional diagnostic information can be found in the log files. Check that the plug-in realm class is in WebSphere Application Server's classpath and follow the realm documentation to ensure that all realm settings in the PaymentServlet init parameters have been specified correctly. The restart your webserver and WebSphere Application Server and try again.

CEPFW0814 An error occurred while attempting to encrypt the cassette data. Please reference the Java stack trace to find out which cassette the error occurred in.

Severity: Error

Explanation: Unable to encrypt data such as the Registration Form, the Registration Form Answers, the Account Data, or the Account Data Answer, etc.

User Response: Contact your service representative.

CEPFW0815 An error occurred while attempting to decrypt the cassette data. Please reference the Java stack trace to find out which cassette the error occurred in.

Severity: Error

Explanation: Unable to decrypt data such as the Registration Form, the Registration Form Answers, the Account Data, or the Account Data Answer, etc.

User Response: Contact your service representative.

CEPFW0816 The WebSphere Commerce Payments is using realm *realmName* which does not have an associated properties file in the Websphere classpath.

Severity: Warning

Explanation: WebSphere Commerce Payments is configured to use a realm for which there is no associated realm properties file. This may not be an error if the realm does not require a properties file. If a properties file is required, then WebSphere Commerce Payments will either not start successfully or you will encounter user access problems.

User Response: If WebSphere Commerce Payments will not start due to a realm error, check that there is a file called (realmName).properties in the Websphere classpath and restart the system. Check the realm documentation to understand whether the realm requires a properties file or not.

CEPFW0817 You must specify a password.

Severity: Error

Explanation: WebSphere Commerce Payments requires a password in order to encrypt sensitive data on the database. You must supply this password when you start WebSphere Commerce Payments. The WebSphere Commerce Payments password is usually the same as your WebSphere Commerce Payments database password.

User Response: Enter the correct WebSphere Commerce Payments password.

CEPFW0818 The specified password was incorrect.

Severity: Error

Explanation: WebSphere Commerce Payments requires a password in order to encrypt sensitive data on the database. You must supply this password when you start WebSphere Commerce Payments. The WebSphere Commerce Payments password is usually the same as your WebSphere Commerce Payments database password.

User Response: Enter the correct WebSphere Commerce Payments password.

CEPFW0819 Could not communicate with the WebSphere Commerce Payments application server ({0} at port {1}).

Severity: Error

Explanation: The IBMPayServer and StopIBMPayServer programs use CAL to communicate with the WebSphere Commerce Payments Application Server. This error results from not being able to send HTTP requests to the PaymentServlet servlet.

User Response: Check that WebSphere Application Server is running and use the Administrator's console to check that the WebSphere Commerce Payments Application Server is running. If you have protected the PaymentServlet servlet with WebSphere security, then you will need to use the -wasuid <userid> and -waspw <password> options to send requests to PaymentServlet. If your WebSphere Commerce Payments application server is listening on a virtual host other than port 80 on the current machine, then you should use the -pmhost <hostname> and -pmport <port> options to contact WebSphere Commerce Payments.

CEPFW0821 IBM WebSphere WebSphere Commerce Payments did not start successfully. Primary return code: {0}. Secondary return code: {1}.

Severity: Error

Explanation: WebSphere Commerce Payments start up processing did not complete successfully.

User Response: Look in the Programmer's Guide to determine the explanation for the combination of primary and secondary return codes. You can use the WebSphere Administrator's Console to set the wpm.initialTrace option in the PaymentServlet Init Parameters to -1 to enable startup tracing and look in the trace logs to further identify the cause of the error.

CEPFW0822 IBM WebSphere WebSphere Commerce Payments did not stop successfully. Primary return code: {0}. Secondary return code: {1}.

Severity: Error

Explanation: WebSphere Commerce Payments stop processing did not complete successfully.

User Response: Look in the Programmer's Guide to determine the explanation for the combination of primary and secondary return codes. You can also enable WebSphere Commerce Payments tracing and look in the trace logs to further identify the cause of the error.

CEPFW0823 IBM WebSphere WebSphere Commerce Payments has started successfully. Password was changed successfully.

Severity: Error

Explanation: The password change request was successfully processed.

User Response: None.

CEPFW0824 Could not read from password file {0}.

Severity: Error

Explanation: The WebSphere Commerce Payments password could not be determined from the specified file.

User Response: Check that the current user is able to read the specified file and that it contains a line of the form ″DBPassword=<password>″.

CEPFW0825 Could not write to password file {0}.

Severity: Error

Explanation: The WebSphere Commerce Payments password could not be written to the specified file.

User Response: Check that the current user is able to write to the specified file.

CEPFW0826 The specified password was incorrect. If you have recently changed your database password, you should use the -changepassword option to change your WebSphere Commerce Payments password and ensure that the old password you specify is correct.

Severity: Error

Explanation: The WebSphere Commerce Payments password is used to connect to the database as well as to protect any sensitive payment data. This error results from the fact that although the password was correctly use to gain access to the database, it was not the password that protects the sensitive data. This condition usually means that the database password has been changed in the operating system and this change has not been reflected in WebSphere Commerce Payments.

User Response: Use the -changepassword option to change the password in WebSphere Commerce Payments.

CEPFW0830 IBM WebSphere WebSphere Commerce Payments is not currently started.

Severity: Error

Explanation: You tried to perform a command that relies on WebSphere Commerce Payments being started. However, WebSphere Commerce Payments has not yet been initialized.

User Response: Use the IBMPayServer command to start WebSphere Commerce Payments.

CEPFW0831 You can only change the WebSphere Commerce Payments password immediately after stopping and starting the WebSphere Commerce Payments Application Server.

Severity: Error

Explanation: You issued a IBMPaymentServer -changepassword command, but the WebSphere Commerce Payments is not in a valid state to successfully process this command.

User Response: Stop and restart the WebSphere Commerce Payments application server via the WebSphere Administrator's console. Then reissue the IBMPayServer -changepassword command.

Appendix E. Notices

Trademarks

Glossary

This dictionary defines technical terms used in the documentation for Payment Suite products. It includes IBM product terminology and may include selected terms and definitions from:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.
- Internet Request for Comments: 1208, *Glossary of Networking Terms*
- Internet Request for Comments: 1392, *Internet Users' Glossary*
- The *Object-Oriented Interface Design: IBM Common User Access Guidelines* , Carmel, Indiana: Que, 1992.

The most current *IBM Dictionary of Computing* is available on the World Wide Web at http:\\www.ibm.com/networking/nsq/nsqmain.htm.

The following cross-references are used in this dictionary:

Contrast with:
This refers the reader to a term that has an opposed or substantively different meaning.

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

Obsolete term for:
This indicates that the term should not be used and refers the reader to the preferred term.

A

access control. In computer security, the process of ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

account. An account is a relationship between the merchant and the financial institution which processes transactions for that merchant. There can be multiple accounts for each payment cassette.

account name. The name you assign to the account. Its only function is to provide display information in the user interface.

acquirer. In e-commerce, the financial institution (or an agent of the financial institution) that receives from the merchant the financial data relating to a transaction and initiates that data into an interchange system.

Address Verification Service (AVS). Within IBM e-commerce, a credit and debit card scheme used by merchants to authenticate the cardholder. The merchant requests the cardholder's address and uses AVS to confirm that the cardholder is who he says he is.

ADSM. See ADSTAR Distributed Storage Manager.

ADSTAR Distributed Storage Manager (ADSM). An IBM licensed program that provides storage management and data access services in a multi-vendor, distributed computing environment.

applet. An application program, written in the Java programming language, that can be retrieved from a Web server and executed by a Web browser. A reference to an applet appears in the markup for a Web page, in the same way that a reference to a graphics file appears; a browser retrieves an applet in the same way that it retrieves a graphics file. For security reasons, an applet's access rights are limited in two ways: the applet cannot access the file system of the client upon which it is executing, and the applet's communication across the network is limited to the server from which it was downloaded. Contrast with servlet.

approve. Within IBM e-commerce, a WebSphere Commerce Payments verb. A merchant issues this verb to create a Payment object. For cassettes that implement credit card protocols, this verb will likely map to authorization (see authorize). Other cassettes may implement the approval process differently. For IBM WebSphere Commerce Payments Cassette for SET and Cassette for CyberCash, the **approve** verb results in the creation of a Payment object and authorization to ensure that funds are available to cover payment.

approve all. Selects all orders displayed for approval.

approved amount. The amount of the order approved for payment.

approve selected. Selects the orders that you want to create a payment in the approved state for. You must perform a manual deposit on this payment to move it from approved state to deposit state.

asymmetric. In computer security, pertaining to the use of different keys for encryption and decryption.

authentication. (1) In SETCo., the process that seeks to validate identity or to prove the integrity of the information. Authentication in public key systems uses digital signatures. (2) In computer security, verification that a message has not been altered or corrupted. (3) In computer security, a process used to verify the user of an information system or protected resources.

authorization. (1) In SETCo., the process by which a properly appointed person or persons grants permission to perform some action on behalf of an organization. This process assesses transaction risk, confirms that a given transaction does not raise the account holder debt above the account credit limit, and reserves the specified amount of credit. (When a merchant obtains authorization, payment for the authorized amount is guaranteed provided that the merchant followed the rules associated with the authorization process.) (2) In computer security, the right granted to a user to communicate with or make use of a computer system. (T) (3) An access right. (4) The process of granting a user either complete or restricted access to an object, resource, or function.

authorization reversal. In SETCo., a transaction sent when a previous authorization needs to be canceled (that is, a full reversal performed) or decreased (that is, a partial reversal performed). A full reversal will be used when the transaction cannot be completed, such as when the cardholder cancels the order or the merchant discovers that goods are no longer available, as when discontinued. A partial reversal will be used when the authorization was for the entire order and some of the goods cannot be shipped, resulting in a split shipment.

authorize. In the credit card world, a merchant is guaranteed that cardholder funds are available to cover a transaction by first *authorizing* the transaction. The cardholder's issuer (that is, the bank that issued the card) guarantees payment.

B

balance. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. Indicates whether the merchant and financial institution agreed on the contents of the batch when it was closed. See 150 for more details.

balanced. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. The batch has been successfully balanced. All totals agree.

balance status. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. The balance status of a batch can be balanced or out of balance.

baseline. In SETCo., a baseline product is the specific product within an operating system family that is run against the SET Tests. A vendor must designate a distinct baseline product for each unrelated operating system family. Refer to the *SET Testing Policies and Procedures* for a complete explanation.

batch. (1) In the credit card world , a batch is a collection of fund transfer requests that are all done at the same time (that is, *in a batch*). Individual fund transfers are not performed for each individual payment, but, rather, a group of transfers is processed so that both the merchant and the financial institution can agree on the funds that are to be transferred. Before a batch is *closed* (that is, the funds are exchanged) there is usually some type of reconciliation process so that both sides agree on the amounts. A group of records or data processing jobs brought together for processing or transmission. (2) Within IBM e-commerce, one of the fundamental WebSphere Commerce Payments objects is the Batch. A Batch is an object with which Payment and Credit objects are associated. Transfer of funds is to occur when the batch is closed. (3) A group of records or data processing jobs brought together for processing or transmission.

batch close date. One of two numeric search parameters that defines the chronological start of your search. Specify a date that precedes the batch close date for the batch you want to search.

batch number. The number that identifies the batch. WebSphere Commerce Payments assigns a number to the batch when the payment is deposited.

batch open date. One of two numeric search parameters that defines the chronological start of your search. Specify a date that precedes the batch open date for the batch you want to search.

batch number. The number that identifies the batch. The number WebSphere Commerce Payments assigns to the batch when the payment is deposited.

batch search. Search for a single batch or group of batches, based on a defined list of characteristics.

BCD. See binary-coded decimal notation.

big endian. A format for storage or transmission of binary data in which the most significant bit (or byte) is placed first. Contrast with little endian.

binary-coded decimal (BCD) notation. A binary-coded notation in which each of the decimal digits is represented by a binary numeral; for example, in binary-coded decimal notation that uses the weights 8, 4, 2, 1, the number “twenty-three” is represented by 0010 0011 (compare its representation 10111 in the pure binary numeration system). (I) (A)

bitmapped message. A variable-length transaction in which each bit in an array of bits indicates the presence or absence of a data field within the transaction.

brand. Within IBM e-commerce, the Cassette object for all of the WebSphere Commerce Payments cassettes (for example, Cassette for SET and Cassette for CyberCash). Each financial transaction for a WebSphere Commerce Payments cassette is associated with a particular brand (for example, MasterCard or VISA). Each account with a financial institution can be configured to support one or more brands.

browser. See Web browser.

browser plug-in. See Web browser plug-in.

C

CA. See certificate authority.

capture. (1) In SETCo., a transaction sent after the merchant has shipped the goods. This transaction will trigger the movement of funds from the Issuer to the Acquirer and then to the merchant account. (2) In the credit card world, payment is actually made when the funds are *captured*. All payments must be authorized *and* captured (although this work can be done at the same time). Note that all payments are associated with a batch (see “void payment” on page 151) and that the actual capture occurs when the associated batch is closed.

capture reversal. In SETCo., a transaction sent when the information in a previous capture message was incorrect or should never have been sent (such as when the goods were not actually shipped). If the capture reversal is the result of incorrect information, it will be followed by a new capture message with the correct information.

cardholder. In e-commerce, a person who has a valid payment card account and uses software that supports e-commerce.

cardholder application. In SETCo., a cardholder application, sometimes called a wallet, that is run by an online consumer that enables secure payment card transactions over a network. SET Cardholder applications must generate SET protocol messages that can be accepted by SET Merchant, Payment Gateway, and Certificate Authority components.

cascading. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which the cluster node with the highest priority for a particular resource acquires the resource if the primary node fails but relinquishes the resource to the primary node upon reintegration of the primary node into the cluster. Contrast with concurrent and rotating.

cassette. (1) In e-commerce, a software component consisting of a collection of Java classes and interfaces that can be easily installed into other software components involved in e-commerce to extend the function of these components. (2) In IBM e-commerce, a WebSphere Commerce Payments concept. The WebSphere Commerce Payments provides a framework that can support many different forms of payment. WebSphere Commerce Payments cassettes are written by IBM or third-party vendors to support different payment protocols (such as, SET and CyberCash) within the WebSphere Commerce Payments Framework. Thus, WebSphere Commerce Payments is an extensible product that can support additional protocols.

cast. In programming languages, an operator that converts the value of its operand to a specified type.

CERN. Conseil Européen pour la Recherche Nucléaire (European Laboratory for Particle Physics). Located in Geneva, Switzerland, CERN initiated the World Wide Web and was the first organization to create a Web server. The CERN Web server is the basis for many commercially available servers.

certificate. (1) In e-commerce, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority (CA). (2) In SETCo., a certificate that has been digitally signed by a trusted authority (usually the cardholder financial institution) to identify the user of the public key. SET defines the following certificate types:

- signature
- key encipherment
- certificate signature
- CRL signature

certificate authority (CA). (1) In e-commerce, an organization that issues certificates. The CA authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. See issuer. (2) In SETCo., certificate authority refers to both the component and to the entity issuing and verifying digital certificates. The component is a product run by a Certificate Authority entity that is authorized to issue and verify digital certificates as requested by Cardholder Wallet components, Merchant Server components, and/or Payment Gateway components over public and private networks.

certificate chain. (1) In SETCo., a hierarchy of digital certificates. The certificate at the top of the hierarchy is called the ″root certificate.″ (2) In SETCo., an ordered grouping of digital certificates, including the root certificate, that are used to validate a specific certificate.

certificate renewal. In SETCo., the process by which a new certificate is created for an existing public key.

certificate revocation. In SETCo., the process of revoking an otherwise valid certificate by the entity that issued the certificate.

certificate revocation list. In SETCo., a list of certificate serial numbers previously issued by a certificate authority that indicate the certificates that are invalid prior to normal expiration due to compromise, disaffiliation, or some other unusual circumstance.

certification. In SETCo., the process of ascertaining that a set of requirements or criteria has been fulfilled and attesting to that fact to others, usually with some written instrument. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

certification authority. See certificate authority.

certified. In SETCo., the process of ascertaining that a set of requirements or criteria has been fulfilled and attesting to that fact to others, usually with some written instrument. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

CGI. See Common Gateway Interface.

CGI program. A computer program that runs on a Web server and uses the Common Gateway Interface (CGI) to perform tasks that are not usually done by a Web server (for example, database access and form processing). A CGI script is a CGI program that is written in a scripting language such as Perl.

CGI script. See CGI program.

clerk. (1) In IBM e-commerce, this is a WebSphere Commerce Payments concept. The WebSphere Commerce Payments has four different access rights. A clerk is defined on a per-merchant basis and has the lowest level of access. (2) A clerk is a low-level employee.

client. A computer system or process that requests a service of another computer system or process that is typically referred to as a server. Multiple clients may share access to a common server.

closed. An order moves into closed state when its associated payment, or payments, moves from deposited state into closed state (that is, when the batch associated with the payment closes). When an order is in closed state, the financial transaction is complete; monies are deposited, and the order cannot be modified. No commands are permitted for orders in this state.

cluster. In high-availability cluster multiprocessing (HACMP), a set of independent systems (called nodes) that are organized into a network for the purpose of sharing resources and communicating with each other.

cluster node. In high-availability cluster multiprocessing (HACMP), an RS/6000 system that participates in a cluster.

commerce service provider (CSP). An Internet service provider that hosts merchant shopping sites and processes payments for the merchants.

Common Gateway Interface (CGI). A standard for the exchange of information between a Web server and computer programs that are external to it. The external programs can be written in any programming language that is supported by the operating system on which the Web server is running. See CGI program.

concurrent. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which all cluster nodes use a resource simultaneously. A cluster node can fail and reintegrate into the cluster without affecting other cluster nodes or the resource. Contrast with cascading and rotating.

confidentiality. In SETCo., the protection of sensitive and personal information from unintentional and intentional attacks and disclosure.

constructor. In programming languages, a method that has the same name as a class and is used to create and initialize objects of that class.

constructor method. See constructor.

conversation. A logical connection between two transaction programs using an LU 6.2 session. Conversations are delimited by brackets to gain exclusive use of a session.

credit. In SETCo., a transaction sent when the merchant needs to return money to the cardholder (via the Acquirer and the Issuer) following a valid capture message, such as when goods have been returned or were defective.

credit reversal. In SETCo., a transaction sent when the information in a previous credit transaction was incorrect or should have never been sent.

cryptographic key. In SETCo., a value which is used to control a cryptographic process, such as encryption or authentication. Knowledge of an appropriate key allows correct decryption or validation of a message.

cryptography. (1) In SETCo., a mathematical process used for encryption or authentication information. (2) The discipline which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and unauthorized use, or a combination thereof. (3) The transformation of data to conceal its contents and to prevent one person from forging or modifying another person's messages.

CSP. See commerce service provider.

CyberCash CashRegister. An electronic payment processing service that is provided by CyberCash, Inc. The CyberCash CashRegister enables merchants to accept and process various types of electronic payments for goods or services that are purchased over the Internet.

CyberCash cassette. A payment cassette that provides support for the CyberCash CashRegister.

D

daily batch totals. The Daily Batch Totals report computes the totals for all batches closed on the date specified on the Search window. The totals include all payments and credits made for all payment types.

decryption. In computer security, the process of transforming encoded text or ciphertext into plain text.

derived products. In SETCo., derived products are components that are created from a product that has received a SET Mark license. Derived products must be created from a product that has received the SET Mark, regardless of operating system family. Please refer to the *SET Testing Policies and Procedures* for a complete explanation.

deposit all . Selects all of the order payments displayed for deposit.

deposited amount . The amount deposited for a Payment. The deposited amount can be different than the approved amount.

deposit selected . Selects the order payments that you want to deposit.

digital envelope. (1) In SETCo., a cryptographic technique to encrypt data and send the encryption key along with the data. Generally, a symmetric algorithm is used to encrypt the data and an asymmetric algorithm is used to encrypt the encryption key. (2) In e-commerce, a package of encrypted data and the encryption key.

digital signature. (1) In SETCo., information encrypted with an entity private key, which is appended to a message to assure the recipient of the authenticity and integrity of the message. The digital signature proves that the message was signed by the entity owning, or having access to, the private key. (2) In e-commerce, data that is appended to, or is a cryptographic transformation of, a data unit and that enables the recipient of the data unit to verify the source and integrity of the unit and to recognize potential forgery.

distinguished name. In SET programs, information that uniquely identifies the owner of a certificate.

document type definition (DTD). The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation may be used within the particular class of documents. A DTD is analogous to a database schema in that the DTD completely describes the structure for a particular markup language.

DTD. See document type definition.

dual signature. In SETCo., a digital signature that covers two or more data structures by including secure hashes or each data structure in a single encrypted block. Dual signing is done for efficiency, that is, to reduce the number of public key encryption operations.

E

EAR file. An Enterprise Archive file represents a J2EE application that can be deployed in a WebSphere application server. EAR files are standard Java archive files and have the file extension .ear.

e-business. Either (a) the transaction of business over an electronic medium such as the Internet or (b) any organization (for example, commercial, industrial, nonprofit, educational, or governmental) that transacts its business over an electronic medium such as the Internet. An e-business combines the resources of traditional information systems with the vast reach of an electronic medium such as the Internet (including the World Wide Web, intranets, and extranets); it connects critical business systems directly to critical business constituencies--customers, employees, and suppliers. The key to becoming an e-business is building a transaction-based Web site in which all core business processes (especially all processes that require a dynamic and interactive flow of information) are put online to improve service, cut costs, and sell products.

ECML. See Electronic Commerce Modeling Language.

e-commerce. (1) The exchange of goods and services for payment between the cardholder and merchant when some or all of the transaction is performed via electronic communication. (2) The subset of e-business that involves the exchange of money for goods or services purchased over an electronic medium such as the Internet.

electronic commerce. See e-commerce.

Electronic Commerce Modeling Language (ECML). In e-commerce, a universal format for wallets that streamlines the collection of electronic data for shipping, billing, and payment on a merchant's Web site and thereby enhances the online shopping experience for consumers and merchants. IBM is one of many companies that are collaborating to develop ECML.

encryption. (1) In SETCo., the process of converting information in order to render it into a form unintelligible to all except holders of a specific cryptographic key. Use of encryption protects information between the encryption process and the decryption process (that is, the inverse of encryption), against unauthorized disclosure. (2) In computer security, the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process.

event. In the Tivoli environment, any significant change in the state of a system resource, network resource, or network application. An event can be generated for a problem, for the resolution of a problem, or for the successful completion of a task. Examples of events are: the normal starting and stopping of a process, the abnormal termination of a process, and the malfunctioning of a server.

event listener. In IBM e-commerce, a computer program that waits to be informed of events of interest and acts upon them.

expiry. (1) The certificate expiration date assigned when the certificate was obtained. Certificates are specific to payment types (for example, SET or CyberCash.) (2) Specifies the card expiration date. An expiry value is required for SET protocol. The value is specified as a string and is used on the payment initiation message. For example, 199911 is an expiry value.

Extensible Markup Language (XML). A standard metalanguage for defining markup languages that was derived from and is a subset of SGML. XML omits the more complex and less-used parts of SGML and makes it much easier to (a) write applications to handle document types, (b) author and manage structured information, and (c) transmit and share structured information across diverse computing systems. The use of XML does not require the robust applications and processing that is necessary for SGML. XML is being developed under the auspices of the World Wide Web Consortium (W3C).

F

failover. See fallover.

fallover. In high-availability cluster multiprocessing (HACMP), an active node's acquisition of resources that were previously owned by another cluster node in order to maintain the availability of those resources.

financial institution. (1) In SETCo., an establishment responsible for facilitating customer-initiated transactions or transmissions of funds for the extension of credit or the custody, loan, exchange, or issuance of money, such as a bank or its designate. (2) Within IBM e-commerce, banks, building societies, and credit unions are examples of financial institutions. An institution that provides financial services.

financial network. Within IBM e-commerce, the aggregate of card processors, acquirers, card issuers, and other institutions through which payment card transaction processing is traditionally performed.

firewall. In communication, a functional unit that protects and controls the connection of one network to other networks. The firewall (a) prevents unwanted or unauthorized communication traffic from entering the protected network and (b) allows only selected communication traffic to leave the protected network.

force. Within IBM e-commerce, a WebSphere Commerce Payments verb. An attempt to settle a batch (see 150). If the reconciliation step fails, the batch is still not closed on the WebSphere Commerce Payments (although it may be out of balance or not closed at the financial institution).

FQDN. See fully qualified domain name.

fully qualified domain name (FQDN). In the Internet suite of protocols, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is `mycomputer.city.company.com`. See host name.

H

HACMP. See high-availability cluster multiprocessing.

handle. In the AIX operating system, a data structure that is a temporary local identifier for an object. Allocating a handle creates it. Binding a handle makes it identify an object at a specific location.

hardware token. In SETCo., a portable device (for example, smart card, PCMCIA cards) specifically designed to store cryptographic information and possibly perform cryptographic functions in a secure manner.

has been certified. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

hash. See root key hash.

heartbeat. In software products, a signal that one entity sends to another to convey that it is still active.

high-availability cluster multiprocessing (HACMP). An application service that enables up to eight RS/6000 servers to access the same data in parallel. This optimizes application execution and scalability and protects against unplanned outages and server downtime.

home page. The initial Web page that is returned by a Web site when a user specifies the uniform resource locator (URL) for the Web site. For example, if a user specifies the URL for the IBM Web site, which is `http://www.ibm.com` , the Web page that is returned is the IBM home page. Essentially, the home page is the entry point for accessing the contents of the Web site. The home page may sometimes be called the ″welcome page″ or the ″front page.″

host. To provide the software and services for managing a Web site.

HostCapture/PostAuth. Within IBM e-commerce, this is a CyberCash concept. One of the three processing models supported by the CyberCash CashRegister service. In particular, the AcquirerProfile field of an account may be set to `HostCapture/PostAuth = 2`, which indicates that the acquirer controls batch processing and a separate deposit request is required to capture the funds after a payment is authorized.

host byte order. The byte order that a central processing unit (CPU) uses to store and process data. This byte order can be big endian or little endian, depending on the particular CPU. Contrast with network byte order.

host name. In the Internet suite of protocols, the name given to a computer. Sometimes, “host name” is used to mean fully qualified domain name; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if `mycomputer.city.company.com` is the fully qualified domain name, either of the following may be considered the host name:

- `mycomputer.city.company.com`
- `mycomputer`

HTML. See Hypertext Markup Language.

HTTP. See Hypertext Transfer Protocol.

Hypertext Markup Language (HTML). A markup language that conforms to the SGML standard and was designed primarily to support the online display of textual and graphical information that includes hypertext links.

Hypertext Transfer Protocol (HTTP). In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

I

idempotency. (1) A property of a mathematical operation whereby repeating the operation produces no change in the final result. For example, the operation of deducting $25.00 from an account balance is not idempotent, but the operation of setting an account balance to $500.00 is idempotent. (2) In SET Secure Electronic Transaction, a property that enables the sender of a request to repeat the request with a guarantee that the outcome will be the same regardless of whether the request is lost, the response is lost, or the request or response is delayed due to network problems. Idempotency is necessary because the SET protocol works in environments where message delivery is not guaranteed, and when the sender does not receive a response, it cannot determine the cause of the delay. If a SET application does not receive a response in a reasonable amount of time, it resends the message; when the receiving SET application determines that it has already processed that message, it retrieves the previous response and sends that response again.

instrumentation. In application or system software, either (a) monitoring functions that provide performance and other information to a management system or (b) the use of monitoring functions to provide performance and other information to a management system.

identify. To establish the identity.

installment payments. In SETCo., a type of payment transaction negotiated between the merchant and the cardholder which permits the merchant to process multiple authorizations. Cardholder specifies a maximum number of permitted Authorization for paying through installment payments.

integrity. In SETCo., the quality of information or a process that is free from error, whether induced accidentally or intentionally.

interactive. In SETCo., a generic class for a network transport mechanism that is dependent on a logical session being maintained during the message exchange (for example, World Wide Web sessions).

internet. (1) In SETCo., the largest collection of networks in the world, interconnected in such a way as to allow them to function as a single virtual network. (2) A collection of interconnected networks that use the Internet suite of protocols. The internet that allows universal access is referred to as the Internet (with a capital "I"). An internet that provides restricted access (for example, to a particular enterprise or organization) is frequently called an intranet, whether or not it also connects to the public Internet.

Internet. The worldwide collection of interconnected networks that use the Internet suite of protocols and permit public access.

interoperability. In SETCo., the ability to exchange messages and keys, both manually and in an automated environment, with any other party implementing this standard, provided that both implementations use compatible options of this standard and compatible communications facilities.

interprocess communication (IPC). The process by which programs communicate data to each other and synchronize their activities. Semaphores, signals, and internal message queues are common methods of interprocess communication.

intranet. A private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

IP address. The unique 32-bit address that specifies the location of each device or workstation on the Internet. For example, 9.67.97.103 is an IP address.

IPC. See interprocess communication.

issuer. (1) In SETCo., the financial institution or its agent that issues the unique primary account number (PAN) to the cardholder for the payment card brand. (2) In e-commerce, a financial institution that issues payment cards to individuals. An issuer can act as its own certificate authority (CA) or can contract with a third party for the service.

J

J2EE. (Java 2 Enterprise Edition) J2EE is designed to support applications that implement enterprise services for customers, employees, suppliers, partners, and others who make demands on or contributions to the enterprise. This can be a single module or a group of modules packaged into an .ear file with a J2EE application deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers.

Java. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

Java Database Connectivity (JDBC). An application programming interface (API) that has the same characteristics as Open Database Connectivity (ODBC) but is specifically designed for use by Java database applications. Also, for databases that do not have a JDBC driver, JDBC includes a JDBC to ODBC bridge, which is a mechanism for converting JDBC to ODBC; it presents the JDBC API to Java database applications and converts this to ODBC. JDBC was developed by Sun Microsystems, Inc. and various partners and vendors.

Java Development Kit (JDK). A software package that can be used to write, compile, debug, and run Java applets and applications.

Java Runtime Environment (JRE). A subset of the Java Development Kit (JDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java Virtual Machine, core classes, and supporting files.

Java Virtual Machine (JVM). A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

JDBC. See Java Database Connectivity.

JDK. See Java Development Kit.

JRE. See Java Runtime Environment.

JVM. See Java Virtual Machine.

K

keepalive message. In Internet communications, a message sent among nodes when no data traffic has been detected for a given period of time. This communication ensures the vitality of the session by keeping the link "alive."

key. In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See private key and public key.

key pair. In computer security, a public key and a private key. When the key pair is used for encryption, the sender uses the public key to encrypt the message, and the recipient uses the private key to decrypt the message. When the key pair is used for signing, the signer uses the private key to encrypt a representation of the message, and the recipient uses the public key to decrypt the representation of the message for signature verification. See asymmetric and digital signature.

key ring. In computer security, a file that contains public keys, private keys, trusted roots, and certificates.

L

little endian. A format for storage or transmission of binary data in which the least significant bit (or byte) is placed first. Contrast with big endian.

M

memory leak. A condition in which a computer program allocates memory and does not free (or properly free) this memory. If the program continues to run and is not terminated, it uses large amounts of real memory and eventually runs out of memory.

merchant. In SETCo., a seller of goods, services, and/or other information who accepts payment for these items electronically. The merchant may also provide electronic selling services and/or electronic delivery of items for sale.

merchant chargeback. Within IBM e-commerce, when fraud occurs and a merchant is liable for funds not obtained, a financial institution may issue a merchant chargeback, reclaiming funds previously credited to a merchant's account.

merchant server. (1) In SETCo., a Merchant Server component is a product run by an online merchant to process payment card transactions and authorizations. It communicates with the Cardholder Wallet, Payment Gateway, and Certificate Authority components. (2) In e-commerce, a Web server that offers cataloged shopping.

merchant settings. The settings that a merchant has made for a cassette. In the WebSphere Commerce Payments user interface, the Payment System object displays as Merchant Settings.

MIME. See Multipurpose Internet Mail Extensions.

mirroring. In the AIX operating system, the maintenance of more than one copy of stored data to prevent the loss of data.

Multipurpose Internet Mail Extensions (MIME). An Internet standard for identifying the type of object being transferred across the Internet. MIME types include several variants of audio, graphics, and video.

mutex. A mutual exclusion lock. See mutual exclusion mechanism.

mutual exclusion mechanism. In software, a method for preventing two separately executing pieces of code from interfering with each other's use of a particular data object. For example, if one thread is executing a function that modifies a shared data structure, the application may need to prevent other threads from simultaneously attempting to read the data before the modifications are complete.

N

network. In SETCo., a collection of communication and information processing systems that may be shared among several users.

network byte order. The byte order that a network uses to transmit data. In the Internet, this byte order is always big endian. Contrast with host byte order.

node. See cluster node.

normal mode. In the IBM Payment Gateway, the processing scheme in which a batched SET message is presented in its entirety to the customized exits of the Payment Gateway Application. Contrast with supervisor mode.

number of credits. In SETCo., a credit is a transaction sent when the merchant needs to return money to the cardholder (via the Acquirer and the Issuer) following a valid capture message, such as when goods have been returned or were defective. Credits can be for up to the total amount of all payments associated with an Order. There can be zero or more Credits per Order.

number of payments. In SETCo., a payment is a request by the merchant to the financial institution to approve all or part of an order. In many cases, all the money authorized for collection by the order will be collected in a single payment. Some payment systems may allow the money authorized in one order (that is, one set of payment instructions) to be collected in multiple payments, depending on the business model. There can be zero or more payments per order.

O

ODBC. See Open Database Connectivity.

ODBC bridge. See Java Database Connectivity.

Open Database Connectivity (ODBC). A standard application programming interface (API) for accessing data in both relational and nonrelational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group and was developed by Digital Equipment Corporation (DEC), Lotus, Microsoft, and Sybase. Contrast with Java Database Connectivity.

online catalog. In SETCo., shopping on the Internet is simple with online catalogs. Online catalogs are Web pages that display items for sale by an online merchant.

order. An order represents all the instructions and information needed from the consumer (payer) in order for the merchant (payee) to collect money.

order amount. The amount of the order.

order fulfillment. Within IBM e-commerce, merchant systems responsible for shipping or distributing orders for which payment has been received. It is believed that an order fulfillment system would query the WebSphere Commerce Payments to determine what goods are to be shipped.

order search. Search for a single order or group of orders, based on a defined set of characteristics.

out of balance. An unsuccessful attempt was made to balance a batch. All totals do not agree.

P

password. For computer or network security, a specific string of characters entered by a user and authenticated by the system in determining the user's privileges, if any, to access and manipulate the data and operations of the system.

payment. In SETCo., a payment is a request by the merchant to the financial institution to approve all or part of an order. In many cases, all the money authorized for collection by the order will be collected in a single payment. Some payment systems may allow the money authorized in one order (that is, one set of payment instructions) to be collected in multiple payments, depending on the business model.

payment amount. The total payment amount deposited by the merchant for this order.

payment card. (1) In SETCo., a term used to collectively refer to credit cards, debit cards, charge cards, and bank cards issued by a financial institution and which reflects a relationship between the cardholder and the financial institution. (2) In e-commerce, a credit card, debit card, or charge card (a) that is issued by a financial institution and shows a relationship between the cardholder and the financial institution and (b) for which a certificate can be issued from an authenticated certificate authority.

payment cassette. A cassette that implements an electronic payment protocol.

payment gateway. (1) In SETCo., a payment gateway component is a product run by an acquirer or a designated third party that processes merchant authorization and payment messages (including payment instructions from cardholders) and interfaces with private financial networks. (2) In e-commerce, the entity that handles transactions between a merchant and an acquirer.

Payment Gateway Application. In the Payment Gateway Transaction Manager (PGTM), the component that processes SET transactions.

Payment Gateway Transaction Manager (PGTM). In the IBM Payment Gateway, the component that is the application-level routing switch. It provides the protocol-level conversion for managing incoming and outgoing communication, and it provides base services for the intelligent routing of transactions to applications. It manages the communication with merchants and routes transactions among merchants, the Payment Gateway Application, and the acquirer's private financial networks.

payment number. (1) Numeric token. (2) A unique identifier for a particular payment within an order.

payment server. In e-commerce, the electronic equivalent of a cash register that organizes and accepts payment for the goods and services selected for purchase. A payment server uses other components, such as a payment gateway and a payment management system, to complete the financial transactions.

Payment Suite. The brand name for IBM's family of payment products for e-commerce.

PGTM. See Payment Gateway Transaction Manager.

port. In the Internet suite of protocols, a specific logical connector between the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) and a higher-level protocol or application. See well-known port.

port number. In the Internet suite of protocols, the identifier for a logical connector between an application entity and the transport service.

primary account number (PAN). In SETCo., the assigned number that identifies the card issuer and cardholder. This account number is composed of an issuer identification number, an individual account number identification, and an accompanying check digit, as defined by ISO 7812–1985.

private key. (1) In SETCo., a cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity and not made public. This key is used to create digital signatures or to decrypt messages or files. (2) In computer security, a key that is known only to its owner. Contrast with public key. See public key cryptography.

protocol. The meanings of, and the sequencing rules for, requests and responses used for managing a network, transferring data, and synchronizing the states of network components.

public key. (1) In SETCo., a cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity publicly available. It is used to verify signatures that were created with he matched private key. Public keys are also used to encrypt messages or files that can only be decrypted using the matched private key. (2) In computer security, a key that is made available to everyone. Contrast with private key. See public key cryptography.

public key cryptography. In computer security, cryptography in which public keys and private keys are used for encryption and decryption.

purge. Within IBM e-commerce, a WebSphere Commerce Payments verb. To remove all associated Payments and Credits from a Batch object, treating it as if it has just been created.

R

random. In SETCo., a value in a set that has equal probability of being selected from the total population of possibilities and is, hence, unpredictable.

realm. In the WebSphere family of products, a database of users, groups, and access control lists. A user must be defined in a realm to access any resource belonging to that realm.

recurring payments. In SETCo., a type of payment transaction initiated by the cardholder that permits the merchant to process multiple authorizations. There are two kinds of recurring payments:

1. multiple payments for a fixed amount
2. repeated billings

refund. Identifies the Credit amount in the smallest denomination of the particular currency used to place the Order.

registration authority. In SETCo., an independent third-party organization that processes payment card applications for multiple payment card brands and forwards applications to the appropriate financial institutions.

reintegration. In high-availability cluster multiprocessing (HACMP), the actions that occur within the cluster when a component that had previously detached from the cluster returns to the cluster. These actions are controlled by the event scripts and when necessary, by manual intervention.

root certificate. In SETCo., the certificate at the top of the certificate hierarchy. See certificate chain.

root key hash. In SET programs, a hexadecimal value that is used to verify the validity of a root certificate. The hash value is published for a consumer to use when the software does not recognize the root certificate.

rotating. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which the cluster node with the highest priority for a particular resource acquires the resource if the primary node fails and retains the resource even upon reintegration of the primary node into the cluster. Contrast with cascading and concurrent.

RS/6000. A family of workstations and servers based on IBM's POWER architecture. They are primarily designed for running multi-user numerical computing applications that use the AIX operating system.

S

sale. (1) In the credit card world, a sale occurs when a transaction is authorized and marked for capture all at once rather than using a two-step process. (2) Within IBM e-commerce, a WebSphere Commerce Payments user interface verb. It means a simultaneous Approve and Deposit.

sale all. Selects all orders displayed to approve and move the associated payment directly into deposited state. The sale function automatically performs an approve and a deposit on your payment.

sale selected. Selects the orders that you want to approve and move the associated payment directly into deposited state. The sale function automatically performs an approve and a deposit on your payment.

sales transaction. In SETCo., a payment authorization transaction that allows a merchant to authorize a transaction and request payment in a single message to the acquirer.

Secure Electronic Transaction. See SET Secure Electronic Transaction.

Secure Sockets Layer (SSL). (1) In SETCo., Secure Socket Layer (SSL) (developed by Netscape Communications Company) is a standard that encrypts data between a Web browser and a Web server. SSL does not specify what data is sent or encrypted. In an SSL session, all data sent is encrypted. (2) A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

server. In SETCo., a functional unit that provides services to one or more clients over a network. Examples include a file server, a print server, and a mail server.

servlet. An application program, written in the Java programming language, that is executed on a Web server. A reference to a servlet appears in the markup for a Web page, in the same way that a reference to a graphics file appears. The Web server executes the servlet and sends the results of the execution (if there are any) to the Web browser. Contrast with applet.

SET. See SET Secure Electronic Transaction.

SET logo. In SETCo., the SET logo or SET Mark is your assurance that the merchant is using software that has successfully completed the SET Software Certification test.

SET cassette. A payment cassette that provides support for the SET protocol.

SET Secure Electronic Transaction. (1) In SETCo., the SET Secure Electronic Transaction™ protocol is an open industry standard developed for the secure transmission of payment information over the Internet and other electronic networks. (2) A specification for securing payment card transactions over open networks such as the Internet. SET was developed by Visa, MasterCard, IBM, and other technology companies.

settle. Within IBM e-commerce, a WebSphere Commerce Payments verb. An attempt to close a Batch object and transfer funds. As part of the settling procedure, there may be some reconciliation or balancing steps (depending on the cassette and financial institution policy) to ensure that the merchant and financial institution agree on the funds being transferred. If the reconciliation step fails, the batch may remain in an open state.

settle all. Settles all batches displayed. The batches can then submit payments and refunds for processing by a payment processor.

settle batches. Settle batches is used to submit batches (payments and refunds) for processing by a payment processor. You can choose to settle one Batch, or multiple Batches.

settle selected. Settles the batches you selected. The selected batches can then submit payments and refunds for processing by a payment processor.

sibling. In SETCo., sibling products are components which, by virtue of being within the same operating system family, are closely related to baseline products. Siblings must be of the same operating system family as the baseline product from which they were created, with identical functionality. Refer to the *SET Testing Policies and Procedures* for a complete explanation.

SMIT. See System Management Interface Tool.

socket. An endpoint provided by the transport service of a network for communication between processes or application programs.

socks-enabled. Pertaining to TCP/IP software, or to a specific TCP/IP application, that understands the socks protocol. “Socksified” is a slang term for socks-enabled.

socksified. See socks-enabled.

socks protocol. A protocol that enables an application in a secure network to communicate through a firewall via a socks server.

socks port. The port on which the Socks server is listening.

socks server. A circuit-level gateway that provides a secure one-way connection through a firewall to server applications in a nonsecure network.

SSL. See Secure Sockets Layer.

stack. A slang term for the set of protocols that comprise TCP/IP. The preferred term is TCP/IP.

supervisor. Can perform all payment processing functions for the merchant.

supervisor mode. In the IBM Payment Gateway, the processing scheme in which a batched SET message is presented as a series of individual requests to the customized exits of the Payment Gateway Application. Contrast with normal mode.

System Management Interface Tool (SMIT). An interface tool of the AIX operating system for installing, maintaining, configuring, and diagnosing tasks.

T

TEC. See Tivoli Enterprise Console.

terminal capture. Within IBM e-commerce, a CyberCash concept. One of the three processing models supported by the CyberCash CashRegister service. In particular, the AcquirerProfile field of an account may be set to `Terminal Capture = 3` , which indicates that the merchant controls batch processing.

thread. A stream of computer instructions that is in control of a process. A multi-threaded process begins with one stream of instructions (one thread) and may later create other instruction streams to perform tasks.

thread pool. The threads that are being used by or are available to a computer program.

time approved. The date and time that this Payment entry was created.

time opened. The time that the batch was created.

time ordered. The time that the order entry was created.

Tivoli Enterprise Console (TEC). A Tivoli product that collects, processes, and automatically initiates corrective actions for system, application, network, and database events; it is the central control point for events from all sources. The Tivoli Enterprise Console provides a centralized, global view of the network computing environment; it uses distributed event monitors to collect information, a central event server to process information, and distributed event consoles to present information to system administrators.

Tivoli GEM. See Tivoli Global Enterprise Manager.

Tivoli Global Enterprise Manager (Tivoli GEM). A Tivoli product that allows system administrators to graphically monitor, control, and configure applications residing in distributed and host (S/390) environments and to use the concept of business systems management to organize related components, thereby providing a business perspective for management decisions. Tivoli Global Enterprise Manager gives information technology staff a logical view of the computing environment; this view shows, at a glance, the status of the multiple applications that comprise the enterprise's business system, including application components, the relationships among and between components, and the flow of data between the applications. By providing this view from a business perspective, Tivoli Global Enterprise Manager enables system administrators to quickly make determinations about the business impact of any component failure. Addressing technology problems from the business perspective greatly improves the effectiveness of system administrators and provides a higher level of service to users.

Tivoli Inventory. A Tivoli product that enables system administrators to gather hardware and software information for a network computing environment. It scans the managed resources and stores inventory information in the configuration repository.

Tivoli management software. The overall descriptor for software from Tivoli Systems Inc., which includes Tivoli Enterprise software (for systems management in a large organization), Tivoli IT Director (for systems management in a small or medium organization), and Tivoli Cross-Site (for the management of e-commerce systems). Tivoli management software enables organizations to centrally manage their computing resources (including the critical applications that drive business performance and profits) in a simple and straightforward manner.

Tivoli Ready. Pertaining to a product that has passed rigorous product certification testing by Tivoli Systems Inc. to ensure that the product delivers turnkey (or ″out-of-the-box″) integration with Tivoli management software. A product that has passed this certification testing carries the Tivoli Ready logo.

transaction. In SETCo., a sequence of one or more related messages.

trust chain. In SETCo., a synonym for certificate chain. See 142.

trusted root. In the Secure Sockets Layer (SSL), the public key and associated distinguished name of a certificate authority (CA).

U

uniform resource locator (URL). (1) A sequence of characters that represent information resources on a computer or in a network such as the Internet. This sequence of characters includes (a) the abbreviated name of the protocol used to access the information resource and (b) the information used by the protocol to locate the information resource. For example, in the context of the Internet, these are abbreviated names of some protocols used to access various information resources: `http` , `ftp` , `gopher` , `telnet` , and `news` ; and this is the URL for the IBM home page: `http://www.ibm.com` . (2) The address of an item on the World Wide Web. It includes the protocol followed by the fully qualified domain name (sometimes called the host name) and the request. The Web server typically maps the request portion of the URL to a path and file name. For example, if the URL is `http://www.networking.ibm.com/nsg/nsgmain.htm` , the protocol is `http` ; the fully qualified domain name is `www.networking.ibm.com`; and the request is `/nsg/nsgmain.htm`.

URL. See uniform resource locator.

user exit routine. A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language.

V

virtual sales slip. In SETCo., detailed information on a financial transaction which is generated by the merchant's online store and downloaded to your digital wallet. Typical items contained in the virtual sales slip are confirmation of your order, shipping details, tax (if applicable), and total amount of sale.

virtual store. An interactive simulation of a store on the World Wide Web.

void payment. Within IBM e-commerce, a verb meaning to nullify or cancel a payment operation (that is, to make it as if it never happened).

W

wallet. In the IBM Payment Suite, software that enables a user to make approved payments to authenticated merchants over public networks and to manage payment card accounts and purchases.

WAR file. A Web Archive (WAR) file is a Java archive file used to store one or more of the following: servlets; JavaServer Pages (JSP) files; utility classes; static documents (such as HTML files, images and sound); client-side applets, beans and classes; descriptive meta-information. Its standard file extension is .war. WAR files are used to package Web modules.

Web. See World Wide Web.

Web browser. (1) Within IBM e-commerce, software running on the cardholder processing system that provides interface to public data networks. (2) A client program that initiates requests to a Web server and displays the information that the server returns.

Web browser plug-in. In SETCo., software installed on the cardholder's computer used to add functions to the Web browser.

webmaster. The person who is ultimately responsible for managing and maintaining a particular Web site.

Web page. Any document that can be accessed by a uniform resource locator (URL) on the World Wide Web. Contrast with home page.

Web server. A server that is connected to the Internet and is dedicated to serving Web pages.

Web site. A Web server that is managed by a single entity (an organization or an individual) and contains information in hypertext for its users, often including hypertext links to other Web sites. Each Web site has a home page. In a uniform resource locator (URL), the Web site is indicated by the fully qualified domain name. For example, in the URL `http://www.networking.ibm.com/nsg/nsgmain.htm` , the Web site is indicated by `www.networking.ibm.com` , which is the fully qualified domain name.

WebSphere. Pertaining to a family of IBM software products that provide a development and deployment environment for basic Web publishing and for transaction-intensive, enterprise-scale e-business applications.

well-known port. In the Internet suite of protocols, one of a set of preassigned protocol port numbers that address specific functions used by transport-level protocols such as the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). The File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP), for example, use well-known port numbers.

World Wide Web (WWW). A network of servers that contain programs and files. Many of the files contain hypertext links to other documents available through the network.

WWW. See World Wide Web.

X

XML. See Extensible Markup Language.

Numerics

2KP transaction. A SET transaction in which the cardholder messages are unsigned and two key pairs (one for the merchant and one for the payment gateway) are used for encryption.

3KP transaction. A SET transaction in which the cardholder messages are unsigned and three key pairs (one for the merchant, one for the payment gateway, and one for the cardholder) are used for encryption.

Index

Readers' Comments — We'd Like to Hear from You

IBM WebSphere Commerce Payments for Multiplatforms
Administrator's Guide
Version 3.1

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Software Reengineering
Department G7IA / Bldg 503
P.O. Box 12195
Research Triangle Park, NC
27709-9990

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.